Università degli Studi di Pisa

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Specialistica in Scienze Fisiche
Anno Accademico 2008-2009

Tesi di Laurea Specialistica

Rinormalizzazione di entanglement in sistemi quantistici unidimensionali

Relatore:
Prof. Massimo Campostrini

Laureando:
Leonardo Bartoli

Indice

Elenco delle figure

Elenco dei listati

Introduzione

La scoperta e lo sviluppo dell'algoritmo DMRG [20] per i sistemi quantistici a molti corpi, ha permesso di determinare le proprietà di alcuni sistemi in fisica dello stato condensato con una accuratezza non disponibile in precedenza. Originariamente DMRG è stato sviluppato per descrivere sistemi unidimensionali a temperatura zero caratterizzati da interazione a corto raggio, tuttavia la versatilità dell'idea alla base di questo metodo, ovvero costruire il flusso del gruppo di rinormalizzazione in uno spazio di matrici densità opportunamente selezionato, si è rivelata tale da permettere di applicarlo anche in contesti contesti lontani da quello originale [15].

La maggiore difficolà dei metodi che utilizzano stati di prodotti di matrici riguarda il fatto che tutti i calcoli hanno un costo computazionale polinomiale $\mathcal{O}\left(\chi^C\right)$ nel troncamento χ che descrive la dimensione dello spazio di Hilbert efficace. Per sistemi di dimensioni superiori ad uno il grado C è tale da rendere accessibili solo simulazioni effettuate per valori di χ piccoli, dell'ordine di 2 o 3, che non garantiscono una accuratezza adeguata. Similmente per sistemi unidimensionali al punto critico il parametro di troncamento aumenta per ogni iterazione del processo di rinormalizzazione, si necessita quindi di un parametro χ grande, nell'ordine di qualche migliaio, per ottenere risultati soddisfacenti [19]. DMRG inoltre presenta delle difficoltà tecniche nel descrivere sistemi caratterizzati da condizioni al contorno periodiche e quindi a descrivere, ad esempio, sistemi invarianti sotto traslazioni, per i quali sarebbe auspicabile una riduzione nel numero dei parametri necessari a descriverne le proprietà fisiche.

Queste difficoltà illustrano come sia emersa la necessità di estendere i metodi usuali, in modo da riuscire a risolvere una classe piú ampia di problemi.

La Teoria Quantistica dell'Informazione ha messo in luce, da un altro punto di vista, le proprietà dei sistemi di molti corpi evidenziando il ruolo dell'entanglement. La conseguente teoria dell'entanglement si è rivelata essere lo strumento principale nella descrizione di fenomeni come il teletrasporto [1] e nello sviluppo della comunicazione e della computazione quantistica [9]. Recentemente il ruolo dell'entanglement ha permesso di approfondire la comprensione dei sistemi in fisica dello stato condensato [10] e di produrre algoritmi che risolvono, almeno parzialmente, le difficoltà incontrate da DMRG [3] basati sull'idea della rinormalizzazione di entanglement (ER). Tale procedura si prefigge l'obbiettivo di ridurre l'intrecciamento tra i blocchi, prima di effettuare il troncamento, in modo tale da contenere la crescita del parametro χ, osservata ad esempio nello studio di sistemi critici, e induce un Ansatz per sistemi di molti corpi detto MERA. Questo metodo incorpora naturalmente l'invarianza sotto traslazioni, nel qual caso necessita di un costo computazionale lineare nelle dimensioni del sistema, e l'invarianza sotto trasformazioni di scala, che comporta una ulteriore

semplificazione computazionale, permettendo di studiare sistemi critici con un costo indipendente dalle dimensioni del sistema.

Di recente sono state proposte promettenti estensioni del metodo allo studio di sistemi quantistici definiti su reticoli in dimensione due, come il modello di Ising [2, 5] ed il modello di Heisenberg su reticolo kagome [4].

In questo lavoro si è sviluppato un algoritmo di ER per sistemi quantistici unidimensionali utilizzando il linguaggio C++, con l'obbiettivo di ottenere un'implementazione efficiente. Questo è stato possibile grazie ai template di espressioni che hanno permesso di realizzare un'interfaccia che mima la notazione di Einstein, comunemente utilizzata in fisica, mantenendo prestazioni paragonabili a quelle che si ottengono con codice scritto manualmente. La flessibilità di questo approccio ha permesso di migliorare leggermente il costo computazionale necessario ad ottimizzare i tensori che codificano la trasformazione di rinormalizzazione.

La tesi è organizzata come segue, nel capitolo 1 si sono introdotte le idee alla base della rinormalizzazione di entanglement e si è mostrato come queste portino a definire un ansatz per sistemi di molti corpi (MERA). Nel capitolo 2 si è specializzata la discussione ad un particolare schema di ER, quello ternario, per reticoli di spin unidimensionali. Si è supposto che il sistema manifesti invarianza sotto traslazioni, sotto queste ipotesi è possibile ridurre ulteriormente il costo computazionale richiesto dall'algoritmo ed ottenerne una descrizione particolarmente efficiente. Nel capitolo 3 si sono illustrati i risultati ottenuti applicando l'algoritmo al modello di Ising in campo magnetico esterno.

Capitolo 1

Rinormalizzazione di entanglement

Indichiamo con $\mathcal{L}$ un reticolo di spin unidimensionale composto da N siti, indicato con $\mathbb{V}_s$ lo spazio vettoriale di singolo sito s, che supponiamo essere di dimensione finita d, il reticolo $\mathcal{L}$ è quindi rappresentato dallo spazio di Hilbert $\mathbb{V}_\mathcal{L}$:

$$\mathbb{V}_\mathcal{L} = \mathbb{V}_1 \otimes \cdots \otimes \cdots \otimes \mathbb{V}_N = \bigotimes_{s\in\mathcal{L}} \mathbb{V}_s, \tag{1.1}$$

la cui dimensione cresce esponenzialmente con il numero dei siti considerati. Questa caratteristica illustra la maggiore difficoltà incontrata nello studio del problema dei molti corpi in meccanica quantistica, in quanto rende una trattazioni esatta inattuabile.

La rinormalizzazione di entanglement è un procedimento numerico che permette di caratterizzare uno stato puro $|\Psi\rangle \in \mathbb{V}_\mathcal{L}$ o, piú in generale, un sottospazio a basse energie $\mathbb{V}_\mathcal{U} \subset \mathbb{V}_\mathcal{L}$ attraverso una riorganizzazione locale dello spazio di Hilbert, basato sul presupposto che, per la natura locale delle interazioni considerate, alcuni gradi di libertà dello stato fondamentale del sistema considerato possano essere disaccoppiati attraverso trasformazioni locali e, di conseguenza, rimossi senza compromettere la descrizione fisica.

1.1 Rinormalizzazione di entanglement

Introduciamo le idee alla base della rinormalizzazione di entanglement, questa è un'estensione delle usuali tecniche con alcune peculiari differenze. Seguendo quanto fatto da Wilson per risolvere il modello di Kondo a basse temperature [21], vogliamo definire una procedura che produca un nuovo reticolo efficace $\mathcal{L}'$ a partire da quello fisico $\mathcal{L}$. Il reticolo efficace viene determinato con un procedimento iterativo. Il primo passo consiste nell'individuare un insieme di blocchi in $\mathcal{L}$. Un blocco $\mathcal{B} \subset \mathcal{L}$ viene definito come un insieme di siti adiacenti del reticolo. La partizione in blocchi deve essere tale che questi abbiamo intersezione vuota, cioè ciascun sito deve appartenere ad un solo blocco. Lo spazio di Hilbert per il sito efficace s' è ottenuto "setacciando" i gradi di libertà fisici dallo spazio $\mathbb{V}_\mathcal{B}$ associato al blocco $\mathcal{B}$

$$\mathbb{V}_\mathcal{B} = \bigotimes_{s\in\mathcal{B}} \mathbb{V}_\mathcal{V} s, \tag{1.2}$$

questa operazione si riduce a determinare una isometria w che rimuova i gradi di libertà non fisici, ovvero che permetta di selezionare il sottospazio $\mathbb{S}_{\mathcal{B}} \subseteq \mathbb{V}_{\mathcal{B}}$ contenente solo l'informazione rilevante

$$w : \mathbb{V'}_{s'} \mapsto \mathbb{V}_{\mathcal{B}}, \qquad w^{\dagger} w = \mathbb{I}. \tag{1.3}$$

L'isometria w puó essere utilizzata per mappare uno stato puro $|\Psi\rangle \in \mathbb{V}_{\mathcal{L}}$, solitamente appartenente al sottospazio a basse energie, in uno stato a grana grossa $|\Psi'\rangle \in \mathbb{V}_{\mathcal{L}'}$.

L'iterazione del procedimento sopra descritto realizza una trasformazione del gruppo di rinormalizzazione numerica (NRG). La scelta del sottospazio $\mathbb{S}_{\mathcal{B}}$ è essenziale per la riuscita della procedura. In particolare la dimensione η di $\mathbb{S}_{\mathcal{B}}$ deve essere la piú piccola possibile in quanto il costo computazionale necessario a calcolare le osservabili locali cresce polinomialmente con η per le successive iterazioni del processo di granulatura. Inoltre η deve essere sufficientemente grande da permettere una descrizione fedele del sistema: lo stato $|\Psi'\rangle$ deve contenere tutte le informazioni rilevanti che erano presenti nello stato originale $|\Psi\rangle$. La scelta ottimale del sottospazio $\mathbb{S}_{\mathcal{B}}$ è stata identificata da White come una parte fondamentale dell'algoritmo DMRG [20]. Considerata la matrice densità ridotta $\rho^{[\mathcal{B}]}$, relativa al blocco $\mathcal{B}$, dello stato $|\Psi\rangle$, il blocco efficace viene scelto in modo tale che

$$\mathbb{V'}_{s'} \approx \mathbb{S}_{\mathcal{B}} \equiv < |\rho_1\rangle, \ldots, |\rho_\eta\rangle >, \tag{1.4}$$

dove gli stati $|\rho_i\rangle$ corrispondono agli η autovettori piú probabili della matrice densità ridotta $\rho^{[\mathcal{B}]}$, cioè quelli relativi agli autovalori p_i maggiori, e l'intero η è soggetto alla condizione

$$1 - \sum_{i=0}^{\eta} p_i \leq \epsilon \tag{1.5}$$

essendo $\epsilon \ll 1$ un errore prestabilito legato al troncamento.

Osserviamo che il parametro η è legato al grado di intrecciamento dello stato $|\Psi\rangle$ tra il blocco $\mathcal{B}$ ed il resto del sistema $\mathcal{L} - \mathcal{B}$, come caratterizzato dalla decomposizione di Schmidt troncata

$$|\Psi\rangle \approx \sum_{i=1}^{\eta} \sqrt{p_i} |\rho_i\rangle \otimes |\phi_i\rangle, \qquad |\phi_i\rangle \in \mathbb{V}_{\mathcal{L}-\mathcal{B}}$$

la procedura descritta ha buone prestazioni quando è possibile fissare η relativamente piccolo, ovvero dipende dallo quantità di entanglement dello stato che si vuole descrivere.

La rinormalizzazione di entanglement si basa sull'osservazione che in molti sistemi il livello di intrecciamento è organizzato per differenti scale, e possa quindi essere ridotto applicando opportune trasformazioni locali. Questa idea viene tradotta in una deformazione, implementata in termini di una trasformazione unitaria, detta disentangler, dei bordi tra il blocco $\mathcal{B}$ ed il resto del sistema.

Per fissare le idee ci specializziamo al caso di una catena di spin unidimensionale e fissiamo il blocco $\mathcal{B}$ in modo tale che questo contenga solo due siti i e j, figura 1.1. Per poter realizzare la rinormalizzazione di entanglement si deve considerare un'altra coppia di siti q, p contigui rispettivamente a i e j ed,

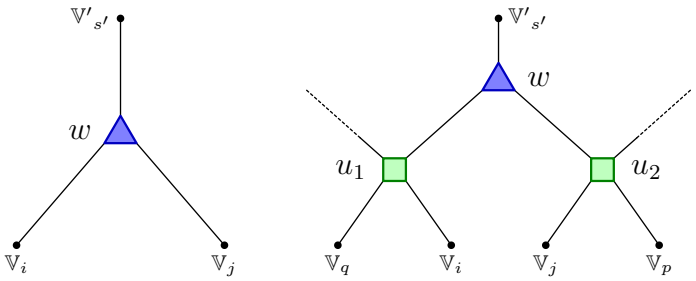


Figura 1.1: in una trasformazione del NRG, ad un blocco di siti viene a corrispondere un sito efficace; la procedura di rinormalizzazione di entanglement aggiunge un passaggio intermedio in quanto al bordo di due blocchi viene prima applicato un operatore unitario con il fine di eliminare l'entanglement a corto raggio.

oltre all'isometria w, una coppia di operatori unitari u_1, u_2 tali che agiscano localmente sulle coppie di siti, ovvero che riorganizzino gli spazi $\mathbb{V}_q \otimes \mathbb{V}_i$, $\mathbb{V}_j \otimes \mathbb{V}_p$:

$$\begin{aligned} u_1 &: \mathbb{V}_q \otimes \mathbb{V}_i \to \mathbb{V}_q \otimes \mathbb{V}_i, \quad u_1^\dagger u_1 = u_1 u_1^\dagger = \mathbb{I}, \\ u_2 &: \mathbb{V}_j \otimes \mathbb{V}_p \to \mathbb{V}_j \otimes \mathbb{V}_p, \quad u_2^\dagger u_2 = u_2 u_2^\dagger = \mathbb{I}. \end{aligned} \tag{1.6}$$

Una scelta opportuna degli operatori puó ridurre l'entanglement tra il blocco $\mathcal{B}$ ed i siti ad esso adiacenti, questa proprietà risulta piú chiara in termini di $\rho^{[\mathcal{B}]}$. La matrice densità ridotta per il blocco $\mathcal{B}$ puó essere ottenuta stracciando la matrice densità relativa al sottospazio contenente i quattro siti q, i, j, p:

$$\rho^{[\mathcal{B}]} = \mathrm{tr}_{qp}\left[\rho^{[qijp]}\right], \tag{1.7}$$

mentre nel caso del NRG l'isometria agiva direttamente su $\rho^{[\mathcal{B}]}$, nel caso della ER questa viene applicata ad una matrice densità riordinata $\tilde{\rho}^{[\mathcal{B}]}$

$$\tilde{\rho}^{[\mathcal{B}]} = \mathrm{tr}_{qp}\left[(u_1 \otimes u_2)\rho^{[qijp]}(u_1 \otimes u_2)^\dagger\right], \tag{1.8}$$

costruita con l'intento di avere un rango efficace $\tilde{\eta}$ minore di quello originario. La trasformazione di ER si compone quindi di due passi:

- partizionato il reticolo in blocchi, ad bordo di ogni blocco si applica una trasformazione unitaria, il disentangler, con l'intento di ridurre l'entanglement con il resto del sistema. Solo l'intrecciamento localizzato al bordo viene rimosso;
- seguendo l'approccio di Wilson, si effettua un troncamento dello spazio di Hilbert di $\mathcal{B}$, il criterio di scelta ottimale è quella introdotto da White applicato alla matrice densità deformata $\tilde{\rho}^{[\mathcal{B}]}$. Questa operazione corrisponde all'applicazione di una trasformazione isometrica, non invertibile.

Iterare il procedimento definisce una successione di reticoli a grana grossa. Ogni passo del processo di rinormalizzazione è implementato da un operatore

isometrico, composizione dei disentangler e delle isometrie, che mappa lo spazio di Hilbert per il reticolo $\mathcal{L}_\theta$ in quello per il reticolo a grana grossa $\mathcal{L}_{\theta+1}$:

$$U_\theta^\dagger \,:\, \mathbb{V}_{\mathcal{L}_\theta} \mapsto \mathbb{V}_{\mathcal{L}_{\theta+1}}. \tag{1.9}$$

Indicato con $\mathcal{L}_0$ il reticolo fisico, l'algoritmo produce infine un reticolo efficace $\mathcal{L}_\Theta$

$$\mathcal{L}_0 \xrightarrow{\text{ER}} \mathcal{L}_1 \xrightarrow{\text{ER}} \cdots \xrightarrow{\text{ER}} \mathcal{L}_\Theta, \tag{1.10}$$

le cui dimensione permettono di calcolare le quantità fisicamente rilevanti con metodi numerici esatti. Analogamente possiamo interpretare la rinormalizzazione in termini dello stato $|\Psi\rangle$ che si vuole descrivere. Posto $|\Psi\rangle \equiv |\Psi_0\rangle$, si viene a determinare una successione di stati a grana grossa

$$|\Psi_0\rangle \xrightarrow{\text{ER}} |\Psi_1\rangle \xrightarrow{\text{ER}} \cdots \xrightarrow{\text{ER}} |\Psi_\Theta\rangle, \tag{1.11}$$

lo stato finale $|\Psi_\Theta\rangle$ deve essere selezionato in modo tale che contenga le informazioni fisicamente rilevanti. Solitamente si vuole descrivere lo stato fondamentale, o un sottopazio a basse energie, del sistema fisico. Se il problema è risolubile quando si consideri il reticolo efficace $\mathcal{L}_\Theta$, è possibile definire la relativa matrice densità ρ_Θ e, invertendo le trasformazioni (1.9) che implementano la ER,

$$|\rho_0\rangle \xleftarrow{\text{ER}^{-1}} |\rho_1\rangle \xleftarrow{\text{ER}^{-1}} \cdots \xleftarrow{\text{ER}^{-1}} |\rho_\Theta\rangle \tag{1.12}$$

ricostruire, almeno parzialmente, l'informazione voluta.

Uno schema di ER è quindi caratterizzato dalla scelta del tipo di partizione del reticolo efficace θ-esimo, dal tipo di disentangler e di isometrie scelti e dal numero di iterazioni effettuate. Gli operatori isometrici sono tensori e il tipo di tensore da considerarsi puó essere espresso mediante una coppia di interi (m, n) dove m indica il numero di siti uscenti, mentre n quello dei siti entranti; ad esempio l'operatore w

$$w : \mathbb{V} \otimes \mathbb{V} \otimes \mathbb{V} \to \mathbb{V}'$$

è del tipo $(1, 3)$, mentre l'operatore u che agisce da disentangler introdotto negli esempi precedenti

$$u : \mathbb{V} \otimes \mathbb{V} \to \mathbb{V} \otimes \mathbb{V}$$

è del tipo $(2, 2)$.

È comodo adottare una notazione diagrammatica che descriva tali scelte. Ad ogni tensore g del tipo (m, n)

$$g : \underbrace{\mathbb{V} \otimes \cdots \otimes \mathbb{V}}_{n \text{ volte}} \to \underbrace{\mathbb{V}' \otimes \cdots \otimes \mathbb{V}'}_{m \text{ volte}}, \tag{1.13}$$

viene associato un opportuno simbolo avente m gambe rivolte verso l'alto ed n gambe verso il basso. Per il tensore $g^\dagger$ si utilizza il simbolo di g riflesso, tale convenzione è illustrata in figura 1.2. Ogni gamba rappresenta un indice che spazia lo spazio di Hilbert locale al quale il tensore viene applicato, gambe connesse rappresentano contrazioni tra gli indici che sono saturati mediante una opportuna δ. Quando i tensori siano isometrie, i vincoli a cui sono soggetti

$$\begin{aligned} g^\dagger g &= \mathbb{I} && \text{se } n \neq m \\ g^\dagger g = g g^\dagger &= \mathbb{I} && \text{se } n = m \end{aligned} \tag{1.14}$$

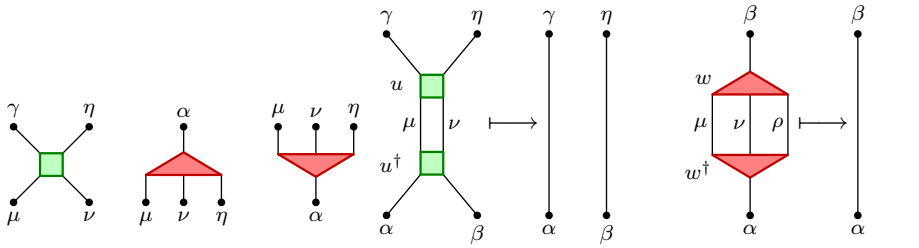


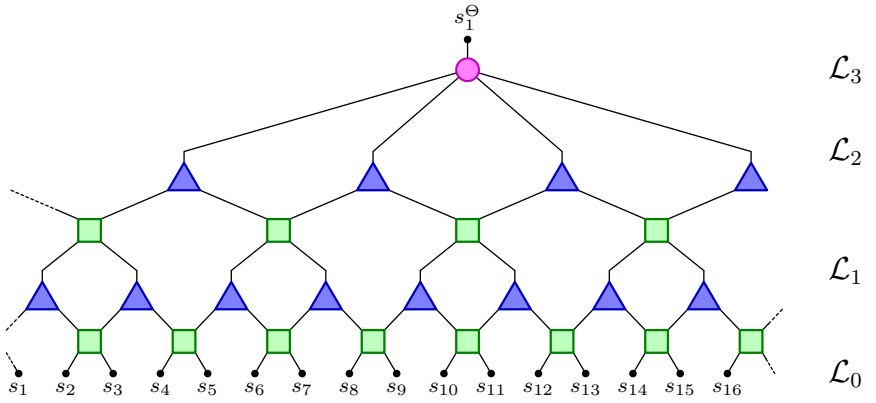
Figura 1.2: (a) ad ogni di tensore viene associato una opportuna rappresentazione diagrammatica, per semplificare la notazione si usa lo stesso simbolo per l'operatore ed il suo hermitiano quando la trasformazione è del tipo (n, n), come nel caso del disentangler u. Segmenti connessi corrispondono a contrazioni dei relativi indici saturati mediante una δ. (b) Rappresentazione grafica dei vincoli a cui sono soggetti i tensori che compongono uno schema ternario.

si traducono nel fatto che ai tensori contratti possano sostituirsi segmenti. I tensori sono organizzati in strati, enumerati da un indice discreto crescente, che "scorrono" la trasformazione di rinormalizzazione e determinano per ogni passo θ-esimo il corrispondente reticolo efficace $\mathcal{L}_\theta$. Per convenzione diciamo l'operatore g della rete di tensori appartiene allo strato θ se appartiene all'insieme delle trasformazioni di rinormalizzazione che mandano lo spazio di Hilbert associato al reticolo $\mathcal{L}_\theta$ nel reticolo $\mathcal{L}_{\theta+1}$. Lo schema di ER per un sistema in D dimensioni è quindi completamente descritto da un opportuno diagramma in $D+1$ dimensioni.

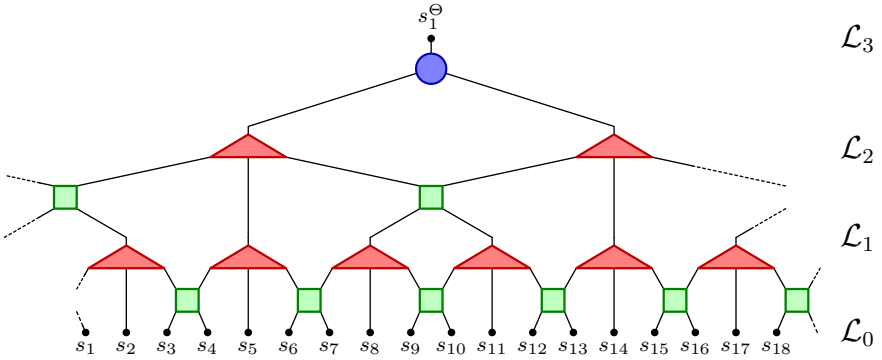
Considerando il caso unidimensionale, sono possibili diverse realizzazioni della rete di tensori. Siamo interessati a studiare quelle reti che presentino una struttura periodica, nel senso che riproducano la medesima prescrizione ad ogni passo del processo di rinormalizzazione.

Schema binario: il reticolo è partizionato in blocchi contenenti ciascuno due siti, si applica quindi un disentangler del tipo $(2, 2)$ e si proietta lo spazio associato a due siti parzialmente disintrecciati in un sito efficace con un'isometria del tipo $(1, 2)$. Il procedimento viene iterato fino a quando non si ottiene un reticolo efficace contenente quattro siti, in questo caso si considera una nuova isometria di tipo $(1, 4)$ che completa la trasformazione di ER. Una rete binaria composta da $\Theta + 1$ strati descrive un reticolo fisico composto da $4 \times 2^\Theta$ siti.

Schema ternario: secondo questa prescrizione, un blocco è formato da tre siti, come nel caso precedente i disentangler che vengono applicati tra i bordi di due blocchi sono del tipo $(2, 2)$ mentre le isometrie associano a tre siti un blocco efficace, sono cioè tensori di tipo $(1, 3)$. Il procedimento viene reiterato fino a quando non si determina un reticolo contenente solo due siti, gli stati fisicamente rilevanti sono fissati mediante un operatore isometrico che chiude la rete e deve quindi essere di tipo $(1, 2)$. Una rete



(a) schema binario per un reticolo di $N = 16$ siti, i tensori utilizzati sono del tipo $(2, 2)$ per i disentangler, del tipo $(1, 2)$ per le isometrie interne e del tipo $(1, 4)$ per il tensore superiore



(b) schema ternario per un reticolo di $N = 18$ siti, il network è organizzato in tre livelli. I tensori utilizzati sono del tipo $(2, 2)$ per i disentangler, del tipo $(1, 3)$ per le isometrie interne e del tipo $(1, 2)$ per il tensore superiore

Figura 1.3: esempi di possibili schemi di ER per reticoli unidimensionali. I tensori che implementano la rinormalizzazione sono organizzati in strati enumerati dal basso verso l'alto in modo tale che un tensore t che contribuisce alla trasformazione $\mathcal{L}_\theta \to \mathcal{L}_{\theta+1}$, sia indicato come t_θ. Osserviamo come ciascuno schema sia, in linea di principio, caratterizzato da un numero di tensori dell'ordine di N, che quindi cresce linearmente con le dimensioni del sistema: sfruttare eventuali simmetrie permette di ridurre notevolmente il numero dei parametri necessari. Adottiamo la convenzione che le linee tratteggiate rappresentino condizioni al contorno periodiche.

ternaria composta da $\Theta + 1$ strati descrive un reticolo fisico composto da $2 \times 3^{\Theta}$ siti.

Da questi esempi possiamo ricavare una proprietà generale della ER, aumentando linearmente i passi del processo di rinormalizzazione, la dimensione del sistema descritto cresce esponenzialmente, in formule

$$\Theta \approx \log(N). \tag{1.15}$$

I due schemi introdotti per i sistemi unidimensionali esibiscono un comportamento diverso per quanto riguarda le proprietà di trasformazione delle osservabili locali. In particolare è piú conveniente utilizzare lo schema binario quando si vogliano trattare operatori locali che agiscano su tre siti contigui mentre quello ternario trova la sua naturale applicazione in sistemi in cui le quantità rilevanti siano operatori che agiscono su due siti primi vicini ed è piú conveniente dal punto di vista computazionale per la particolare struttura diagrammatica che lo caratterizza. Per studiare in dettaglio queste problematiche è conveniente introdurre una corrispondenza tra le reti di tensori associate agli schemi di ER ed i circuiti quantistici.

1.2 Multi-scale entanglement renormalization ansatz

La rinormalizzazione di entanglement porta naturalmente a considerare reti di tensori che possono essere messe in corrispondenza con i circuiti quantistici definiti in teoria quantistica dell'informazione. Questo corrispondenza permette di determinare le proprietà strutturali delle reti e conduce al *multi-scale entanglement renormalization ansatz* (MERA), un ansatz per i sistemi di molti corpi capace di esplorare sistemi quantistici a piccole dimensioni, tipicamente catene di spin in una o due dimensioni, arbitrariamente grandi anche in concomitanza di punti critici quantistici.

Circuiti Quantistici

Il bit è il concetto fondamentale della teoria della teoria dell'informazione classica. La teoria dell'informazione quantistica [9] si basa sull'idea di sostituire al bit il suo analogo quantistico detto *quantum bit* o piú semplicemente *qubit*. Il qubit è essenzialmente uno stato $|q\rangle$ in un sistema quantistico a due livelli $\mathbb{V}_2$, a differenza del caso classico non vi sono particolari restrizioni che impongano l'utilizzo di sistemi a due livelli, possiamo quindi considerare oggetti piú generali detti *qudit* che corrispondono a stati in sistemi quantistici a d livelli. Ad un registro classico corrisponde un registro di qudit, definito come uno stato appartenente al prodotto tensoriale degli spazi di Hilbert $\mathbb{V}_q$ dei qudit che lo compongono:

$$|q_1, \ldots, q_n\rangle = |q\rangle_1 \otimes \cdots \otimes |q\rangle_n, \qquad |q\rangle_i \in \mathbb{V}_q.$$

Le porte logiche fondamentali vengono trasposte in porte quantistiche, ovvero in trasformazioni che agiscono su registri di qudit con la importante differenza che, nel caso quantistico, poiché la probabilità si deve conservare, le operazioni devono essere invertibili, piú precisamente devono essere espresse in termini di

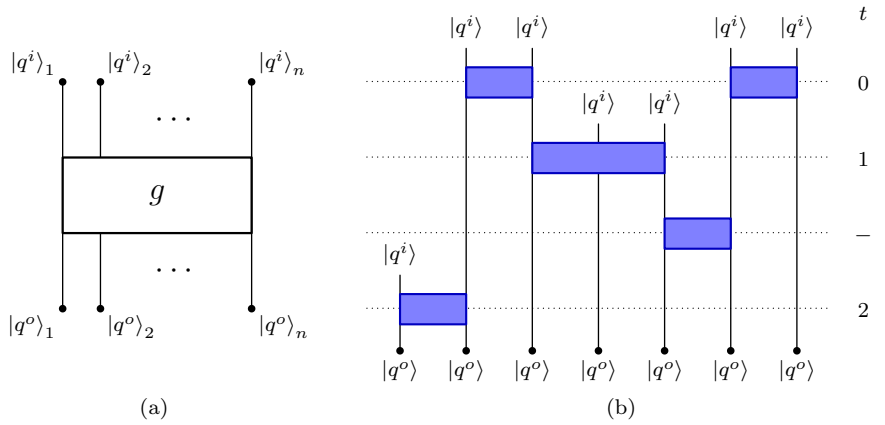


Figura 1.4: (a) la porta logica quantistica g trasforma un registro di qubit, la richiesta che g sia unitaria impone ad un input di n qudit, corrisponda un output dello stesso tipo; (b) esempio di circuito quantistico ordinato cronologicamente.

operatori unitari, ció impone che una porta quantistica g mandi un registro di n qubit in un registro delle stesse dimensioni

$$g : \mathbb{V}_q^{\otimes n} \ni |q_1^i, \ldots, q_n^i\rangle \mapsto |q_1^o, \ldots, q_n^o\rangle \in \mathbb{V}_q^{\otimes n}, \qquad gg^\dagger = g^\dagger g = \mathbb{I}.$$

Dalla definizione segue che ad ogni porta logica quantistica possa essere associato un tensore di tipo (n, n) secondo la convenzione introdotta per la rinormalizzazione di entanglement, in particolare possiamo utilizzare una rappresentazione diagrammatica simile a quella introdotta per la ER in cui le gambe superiori indichino gli ingressi e quelle inferiori le uscite, invertendo quindi l'ordinamento cronologico per ragioni che saranno evidenti in seguito.

Un circuito quantistico è un modello che descrive un calcolo quantistico che associa ad un registro di qudit entrante, o registro di input, ad un registro di qudit uscente, o registro di output ed è la trasposizione di un circuito logico classico. Generalizzando quanto viene fatto nel caso classico i circuiti quantistici sono composti da porte elementari e da fili che le connettono e che rappresentano lo stato di un singolo qubit. Da un punto di vista topologico, i circuiti quantistici ammessi sono un sottoinsieme di quelli classici in quanto devono essere aciclici e non ammettono la congiunzione di due fili, nell'operazione classica detta FANIN, che associa al filo risultante lo XOR bitwise degli input considerati. Tale operazione chiaramente non è invertibile e non è quindi ammessa in un circuito quantistico, cosí come non è direttamente realizzabile l'operazione inversa, detta FANOUT, che copia lo stato associato ad un filo.

Un circuito quantistico $\mathcal{C}$ è ordinato cronologicamente quando le porte logiche e gli ingressi sono organizzati in strati. Tali strati sono enumerati da una variabile discreta che viene incrementata quando nello strato considerato sia presenta almeno un ingresso. Adottiamo quindi la convenzione grafica di ordinare il circuito dall'alto verso il basso come illustrato dalla figura 1.4.

Consideriamo quindi una porta logica g al tempo t di un circuito ordinato $\mathcal{C}$, quando un ingresso di g è connesso all'output di una porta logica presente

ad un tempo t' passato rispetto a t, cioè $t' < t$, il filo corrispondente è detto legato. Se all'ingresso corrisponde direttamente un qudit del registro di input, il filo è detto libero.

Cono causale passato: dato un circuito quantistico ordinato cronologicamente $\mathcal{C}$, definiamo il cono causale passato $\mathcal{C}^{[s]}$ di un qudit uscente $|s\rangle$ come il sottocircuito di $\mathcal{C}$ formato da tutte le porte e da tutti i fili causalmente connessi con s, cioè da quelli che possono modificare lo stato dell'output $|s\rangle$. Per ogni passo temporale t definiamo la larghezza locale di $\mathcal{C}^{[s]}$ al tempo t come la somma del numero di ingressi legati delle porte logiche presenti nella restrizione, a quel passo temporale, del cono causale. La larghezza del cono causale $\mathcal{C}^{[s]}$ è il valore massimo della larghezza locale al variare dell'indice temporale. Dato un insieme di siti $s_1, \ldots, s_n$ il cono causale passato $\mathcal{C}^{[s_1,\ldots,s_n]}$ è l'unione dei coni casuali dei singoli siti $\mathcal{C}^{[s_1]}, \ldots, \mathcal{C}^{[s_n]}$:

$$\mathcal{C}^{[s_1,\ldots,s_n]} = \bigcup_{i=1}^{n} \mathcal{C}^{[s_i]}.$$

ER e circuiti quantistici

Come visto, uno schema di ER definisce in modo naturale una rete di tensori che lo caratterizzano completamente. È possibile interpretare tale rete come un circuito quantistico. Consideriamo inizialmente il ruolo dei disentangler. Tali operatori si prefiggono il compito di rinuovere l'intrecciamento a corto raggio presente nel sistema e possiamo schematizzare questa operazione ponendo che lo stato $|\Psi\rangle$ venga fattorizzato come

$$|\Psi\rangle = |\Psi'\rangle \otimes |\sigma\rangle, \tag{1.16}$$

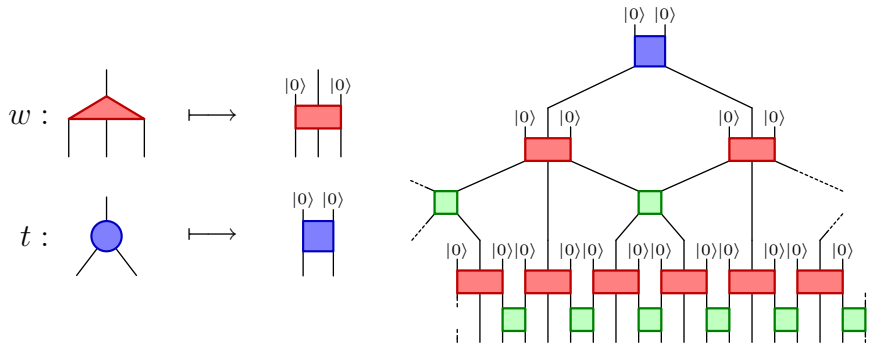
dove la componente $|\sigma\rangle$ viene quindi rimossa mediante l'azione di opportune isometrie. Possiamo quindi formalmente ripristinare i gradi di libertà rimossi imponendo che corrispondano allo stato fondamentale di un opportuno spazio di Hilbert, ovvero associamo alle isometrie dello schema di rinormalizzazione una porta quantistica in cui alcuni ingressi sono in uno stato fissato $|0\rangle$ come illustrato in figura 1.5.

Il circuito quantistico associato ad uno schema di ER ha alcune importanti proprietà. Osserviamo che, per costruzione, l'operatore unitario $U_{\mathcal{C}}$, codificato dal circuito, associa al registro di input completamente disintrecciato $|0\rangle^{\otimes N}$ lo stato rappresentato dalla trasformazione di ER

$$|\Psi\rangle = U_{\mathcal{C}}|0\rangle^{\otimes N}. \tag{1.17}$$

Inoltre presenta naturalmente un ordinamento cronologico inverso rispetto al flusso del gruppo di rinormalizzazione che porta alla seguente definizione

MERA: uno schema per la rinormalizzazione di entanglement produce un ansatz per sistemi quantistici di molti corpi detto *multi-scale entanglement renormalization ansatz* (MERA), quando il circuito quantistico indotto dallo schema ha un cono causale limitato per ogni sito s del reticolo fisico, indipendente dal numero di iterazioni Θ del processo di rinormalizzazione.



(a) agli operatori isometrici vengono associate le rispettive trasformazioni unitarie in cui alcuni ingressi sono in uno stato fissato $|0\rangle$

(b) il circuito quantistico è tale da restituire lo stato $|\Psi\rangle$ quando come input sia considerato lo stato, completamente fattorizzato, $|0\rangle^{\otimes N}$.

Figura 1.5: circuito quantistico $\mathcal{C}$ associato al MERA ternario unidimensionale

Si vuole dimostrare che lo schema ternario costituisce un MERA. Procediamo iterativamente mostrando che tale schema conserva la struttura degli operatori definiti su due siti, come vedremo questa proprietà è strettamente connessa con il fatto che la rete di tensori considerata sia caratterizzata da un cono causale limitato.

Consideriamo una coppia di siti primi vicini (s, r) nel reticolo $\mathcal{L}_\theta$ al tempo t. Ci sono tre possibili scelte che corrispondono alla posione relativa dei siti rispetto al partizionamento in blocchi del reticolo, ovvero i due siti possono appartenere ad uno stesso blocco $\mathcal{B}_s$ o appartengono a due blocchi distinti $\mathcal{B}_s$, $\mathcal{B}_{s+1}$, ma consecutivi. Quale che sia la scelta effettuata il cono causale al tempo t, ha larghezza due e contiene due fili legati contigui, il disentangler impone infatti che, anche quando i due siti appartengano al medesimo blocco, debbano contribuire a due siti s' e r' nella granitura, figura 1.6. Questa proprietà è indipendente dal numero di strati considerati e quindi il cono causale associato a $\mathcal{C}^{[sr]}$ è limitato. In termini di osservabili questo implica che lo schema ternario manda operatori definiti su siti primi vicini in operatori dello stesso tipo.

Consideriamo ora un sito s. Se s è il sito centrale di un blocco $\mathcal{B}_s$ allora deve dare contributo solo al sito s', il suo cono causale $\mathcal{C}^{[s]}$ ha larghezza uno. Le altre due possibili scelte, fissare s a destra o sinistra del sito centrale del blocco $\mathcal{B}_s$, portano ad un risultato analogo a quanto visto nel caso precedente: l'azione del disentangler impone che l'informazione contenuta in s contribuisca ai siti efficaci s' e r'. Il cono causale al tempo t che ne consegue ha larghezza due rispettivamente. Applicare successive trasformazioni, o in termini di teoria dell'informazione, scorrere a ritroso il circuito quantistico permette di determinare che, nella peggiore delle ipotesi il cono causale per un sito s ha larghezza 2, indipendentemente dal numero di passi fatti, che dimostra quanto voluto.

Questa discussione permette di determinare una condizione sufficente per determinare se uno schema di ER costituisce un MERA, basta infatti verificare che esista una struttura minimale conservata, intentendento con questo un insieme finito di n siti primi vicini che producano n siti primi vicini lungo il

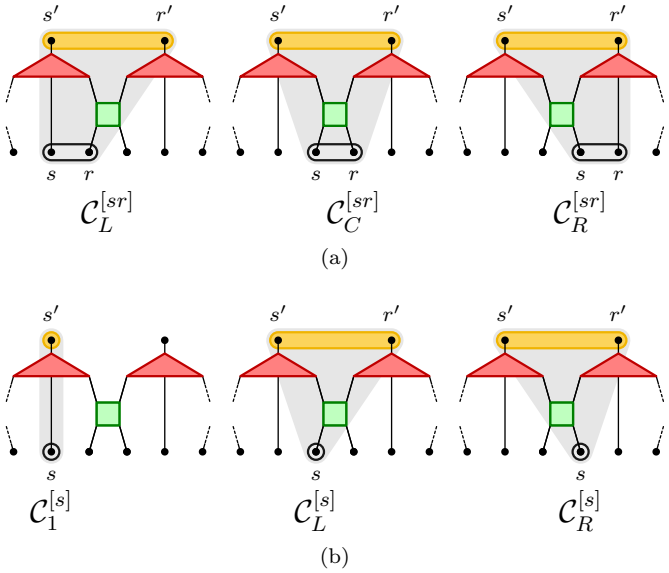


Figura 1.6: cono causale per MERA ternario per una coppia di due siti primi vicini (a) e per un singolo sito (b)

flusso del gruppo di rinormalizzazione, indipendentemente da come vengano scelti.

Nello stesso spirito è facile dimostrare che anche lo schema binario soddisfa le condizioni richieste per un MERA, con l'importante differenza che in questo caso la struttura minima conservata è composta da una tripletta di siti s, r, v: il cono causale è piú largo nel caso binario e questo suggerisce che il costo computazionale associato a tale scelta sia piú elevato di quello che si avrebbe fissando un MERA ternario. In termini di osservabili questo implica che lo schema binario trasforma operatori definiti su tre siti in operatori dello stesso tipo.

Un MERA che descriva un sistema fisico non omogeneo deve essere caratterizzato da $\mathcal{O}(N)$ tensori a meno che non siano presenti simmetrie che permettano di ridurne il numero. I tensori della rete non devono avere necessariamente le medesime dimensioni. In termini dei reticoli efficaci ció significa che la dimensione dello spazio vettoriale $\mathbb{V}_s^{\theta}$, associato al sito $s \in \mathcal{L}_\theta$, dipende dallo strato θ. Definiamo quindi la dimensione di sito per lo strato θ come la quantità

$$\dim(\mathbb{V}_s^{\theta}) \equiv \chi_\theta, \tag{1.18}$$

osserviamo che, per costruzione, $\chi_0 = d$. Solitamente si utilizza una dimensione di sito che, dopo un certo numero di passi, sia indipendente dal parametro θ, chiamiamo tale valore parametro di rifinitura χ del MERA. Spesso esprimeremo il costo computazionale degli algoritmi considerati in termini del parametro di rifinitura.

La peculiare struttura causale richiesta, insieme ai vincoli imposti sulle isometrie (1.14), rendono MERA un metodo molto efficiente per il calcolo dei valori di aspettazione delle osservabili locali.

Simmetrie

La rinormalizzazione di entanglement si presta facilmente ad includere simmetrie spaziali. Consideriano un sistema unidimensionale invariante sotto traslazioni. Per ogni blocco in cui viene partizionato il reticolo dobbiamo risolvere lo stesso problema in quanto i termini hamiltoniani presenti sono tutti identici per costruzione. Questa osservazione porta ad introdurre un MERA invariante sotto traslazioni, ovvero uno schema di rinormalizzazione in cui, per ogni strato, possiamo introdurre una unica coppia di tensori w, u che sono quindi univocamente determinati dall'indice θ, riducendo il numero di parametri all'ordine $\mathcal{O}(\Theta) = \mathcal{O}(\log N)$ (ricordiamo che il numero di siti N è legato al parametro Θ dalla relazione $\Theta \approx \log N$). Osserviamo che un MERA invariante sotto traslazioni non descrive necessariamente uno stato fondamentale, o un sottospazio, che manisfesti tale simmetria poiché la struttura diagrammatica induce delle disomogeneità: alcuni siti hanno posizioni inequivalenti rispetto alla rete di tensori. È comunque possibile mitigare questo effetto, mediando opportunamente le espressioni per le osservabili locali e le matrici densità ridotte.

In sistemi invarianti sotto trasformazioni di scala è possibile ridurre ulteriormente i parametri necessari imponendo che, dopo un certo valore $\bar{\theta}$, tutti i disentangler e le isometrie siano identici. Il numero di parametri risulta quindi indipendente dalle dimensioni fisiche del sistema considerato. Tale scelta definisce un MERA invariante di scale che trova applicazione nello studio di sistemi quantistici in corrispondenza del punto critico o di sistemi caratterizzati da ordine topologico al limite infrarosso del flusso del gruppo di rinormalizzazione [12, 8].

Capitolo 2

MERA Ternario

Studiamo le proprietà di un MERA ternario. In tale schema, a ciascun passo, il reticolo $\mathcal{L}_\theta$ è partizionato in blocchi contenenti tre siti, viene quindi applicato prima un disentangler u del tipo $(2,2)$ quindi un'isometria w del tipo $(1,3)$ per produrre il reticolo $\mathcal{L}_{\theta+1}$. Un operatore è identificato dal passo θ in cui interviene e dai siti ai quali viene applicato in modo non banale. Indicizzati con s e t siti appartenenti rispettivamente al reticolo $\mathcal{L}_\theta$ e $\mathcal{L}_{\theta+1}$, fissate le coordinate per i relativi spazi di Hilbert $\mathbb{V}_{\mathcal{L}_\theta}$ e $\mathbb{V}_{\mathcal{L}_{\theta+1}}$, la trasformazione (1.9) diventa

$$|t_0,\dots,t_m\rangle = w^{*t_0}_{r_0,r_1,r_2}\cdots w^{*t_m}_{r_n,r_0,r_1} u^{*r_2,r_3}_{s_2,s_3}\cdots u^{*r_n,r_0}_{s_n,s_0}|s_0,\dots,s_n\rangle, \tag{2.1}$$

dove i tensori u, w soddisfano i seguenti vincoli

$$\begin{cases} (u)^{r,s}_{t,v}(u^\dagger)^{t,v}_{p,q} = \delta^r_p\delta^s_q, \\ (w)^{p}_{s,t,v}(w^\dagger)^{s,t,v}_{q} = \delta^p_q. \end{cases} \tag{2.2}$$

In vista di determinare come sia possibile trasformare la Hamiltoniana del sistema e costruire le matrici densità ridotte, definiamo gli operatori di salita e di discesa associati ad una rete di tensori.

2.1 Operatori di salita

Una rete di tensori MERA è particolarmente efficace nel descrivere osservabili locali. Consideriamo un osservabile O_θ definito nello spazio vettoriale che descrive il reticolo $\mathcal{L}_\theta$, l'immagine dell'operatore al passo successivo del processo di rinormalizzazione è ottenuta mediante l'applicazione dell'operatore (1.9)

$$O_{\theta+1} = {U_\theta}^\dagger O_\theta U_\theta. \tag{2.3}$$

In genere, per la struttura delle contrazioni rappresentate della rete di tensori, è complicato determinare una forma esplicita pre l'operatore $O_{\theta+1}$. In alcuni casi le espressioni si semplificano notevolmente.

Supposto che O_θ agisca in modo non banale solo su una coppia di siti primi vicini $s, s+1 \in \mathcal{L}_\theta$:

$$O_\theta = \mathbb{I}\otimes\mathbb{I}\cdots\otimes o_\theta^{[s,s+1]}\otimes\cdots\mathbb{I}\otimes\mathbb{I}, \tag{2.4}$$

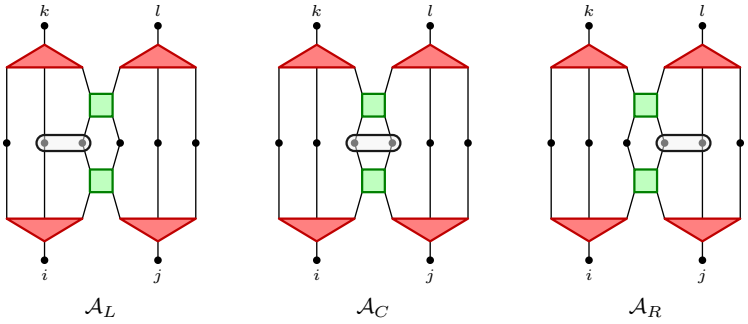


Figura 2.1: esistono tre possibili realizzazioni $\mathcal{A}_L$, $\mathcal{A}_C$, $\mathcal{A}_R$ per l'operatore di salita quando si consideri un operatore che si applichi a due siti primi vicini.

è possibile, ricorrendo ai vincoli (2.2), semplificare tutti i tensori che non appartengono al cono causale della coppia di siti r, $r+1$. L'operatore di salita per l'osservabile O_θ è quindi determinato dalla posizione relativa dei siti s, $s+1$ rispetto ai tensori della rete, figura 2.1. Sono possibili tre configurazioni differenti che definiscono tre diversi operatori di salita $\mathcal{A}^\theta_A$

$$O_{\theta+1} = \mathcal{A}^\theta_A(O_\theta),\ A = \{L, C, R\}. \tag{2.5}$$

Questo risultato si estende immediatamente al caso in cui un osservabile si possa decomporre come somma di termini che agiscono su due siti. Per linearità il risultante operatore di salita risulta uguale alla somma degli operatori di salita applicati ai singoli termini.

Per un osservabile invariante sotto traslazioni è conveninte introdurre l'operatore di salita medio definito come la media dei tre differenti contributi. Un operatore invariante sotto traslazioni è completamente caratterizzato da un singolo termine $o^{[r,r+1]}$, l'operatore di salita medio permette di determinarne iterativamente la sua rappresentazione per ogni spazio di Hilbert $\mathbb{V}_{\mathcal{L}_\theta}$:

$$o_{\theta+1} = \overline{\mathcal{A}}(o_\theta) \equiv \frac{1}{3}\Big[\mathcal{A}^\theta_L(o_\theta) + \mathcal{A}^\theta_C(o_\theta) + \mathcal{A}^\theta_R(o_\theta)\Big]. \tag{2.6}$$

2.2 Operatori di discesa

Siano $r, r+1$ due siti appartenenti al reticolo $\mathcal{L}_\theta$. Vogliamo costruire degli operatori di discesa che mappino la matrice densità $\rho^{[r,r+1]}_\theta$ in una matrice densità locale, definita sui siti $[s, s+1]$ appartenenti al cono causale passato di $[r, r+1]$ nel reticolo $\mathcal{L}_{\theta-1}$. In questo caso la scelta non è univoca, scegliamo di definire gli operatori come illustrato in figura 2.2, in questo modo l'operatore di discesa $\mathcal{D}$ è il duale del relativo operatore di salita. Dato un qualsiasi osservabile $o^{[s,s+1]}_{\theta-1}$ ed una matrice densità $\rho^{[r,r+1]}_\theta$ si ha

$$\mathrm{tr}\Big(o^{[s,s+1]}_{\theta-1}\mathcal{D}(\rho^{[r,r+1]}_\theta)\Big) = \mathrm{tr}\Big(\mathcal{A}(o^{[s,s+1]}_{\theta-1})\rho^{[r,r+1]}_\theta\Big). \tag{2.7}$$

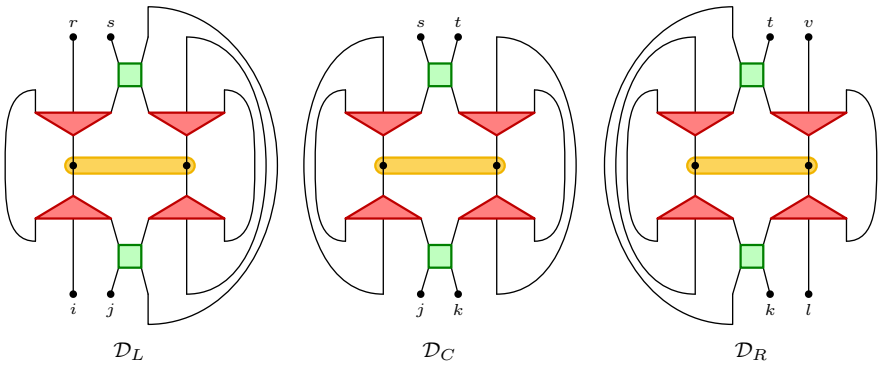


Figura 2.2: gli operatori di discesa sono scelti in modo tale da trasformare una matrice densità locale ρ_θ in una equivalente matrice densità locale $\rho_{\theta-1}$ definita sul reticolo $\mathcal{L}_{\theta-1}$. A seconda della posizione relativa del supporto di $\rho_{\theta-1}$ con il piú vicino disentangler è possibile contrarre i tensori in tre modi diversi corrispondenti alle definizioni degli operatori di discesa $\mathcal{D}_L$, $\mathcal{D}_C$ e $\mathcal{D}_R$.

Come nel caso degli operatori di salita, è utile definire un operatore di discesa medio $\overline{\mathcal{D}}$

$$\rho_{\theta-1} = \overline{\mathcal{D}}(\rho_\theta) \equiv \frac{1}{3}\Big[\mathcal{D}_L^\theta(\rho_\theta) + \mathcal{D}_C^\theta(\rho_\theta) + \mathcal{D}_R^\theta(\rho_\theta)\Big], \tag{2.8}$$

attraverso il quale é possibile costruire una matrice densità che approssimi uno stato, o un sottospazio, invariante sotto traslazioni. L'utilizzo dell'operatore di discesa medio permette infatti di ridurre le anisotropie dovute alla geometria della rete di tensori.

2.3 Hamiltoniana e matrice densità

Dalla struttura degli operatori di salita e di discesa, un MERA ternario si presta a descrivere sistemi la cui Hamiltoniana sia decompomibile in una somma di termini che agiscano su una coppia di siti primi vicini

$$H = \sum_s h^{[s,s+1]}. \tag{2.9}$$

Data una rete di tensori, ciascun termine della (2.9) trasforma attraverso l'applicazione di un opportuno operatore di salita. Per linearità quindi un MERA ternario conserva la struttura dell'interazione lungo il flusso del gruppo di rinormalizzazione

$$H_\theta = \sum_{s\in\mathcal{L}_\theta} h_\theta^{[s,s+1]} = \sum_{s\in\mathcal{L}_\theta} \mathcal{A}(h_{\theta-1}^{[s,s+1]}). \tag{2.10}$$

Per non appesantire la notazione abbiamo sottinteso la diversa forma strutturale dell'operatore di salita applicato al termine hamiltoniano considerato.

La trasformazione di rinormalizzazione produce quindi una successione di Hamiltoniane fino a determinare l'Hamiltoniana H_Θ associata allo strato

superiore di dimensioni tali che sia possibile trattarla con metodi numerici esatti. Determinata quindi calcolare la matrice densità ρ_Θ associata al reticolo $\mathcal{L}_\Theta$, applicando opportunamente una successione di operatori di discesa, è possibile calcolare le matrici densità ridotte di interesse [3].

Per costruzione, un sistema invariante sotto traslazioni è caratterizzata da un solo termine hamiltoniano che possiamo quindi trasformare mediante l'applicazione dell'operatore di salita medio $\overline{\mathcal{A}}$

$$h_o \xrightarrow{\overline{\mathcal{A}}} h_1 \xrightarrow{\overline{\mathcal{A}}} \dots \xrightarrow{\overline{\mathcal{A}}} h_\Theta, \tag{2.11}$$

la matrice densità ridotta per due siti primi vicini che puó essere calcolata mediante l'applicazione dell'operatore di discesa medio $\overline{\mathcal{D}}$

$$\rho_o \xleftarrow{\overline{\mathcal{D}}} \rho_1 \xleftarrow{\overline{\mathcal{D}}} \dots \xleftarrow{\overline{\mathcal{D}}} \rho_\Theta, \tag{2.12}$$

l'energia del sistema è completamente determinata in funzione del termine hamiltoniano h e della matrice densità ridotta ρ.

Visto come sia possibile trasformare le quantità fisicamente rilevanti, è possibile stabilire un algoritmo per ottimizzare la rete di tensori con il fine di descrivere un sottospazio a basse energie $\mathbb{V}_\mathcal{U} \subset \mathbb{V}_\mathcal{L}$ del sistema.

2.4 Ottimizzazione della rete di tensori

Data una rete di tensori che costituiscono un MERA, si pone il problema di ottimizzarne gli elementi per descrivere le proprietà spettrali, a basse energie, del sistema considerato. Per fissare le idee specializziamo la discussione allo schema ternario, il procedimento che verrà introdotto è facilmente estendibile ad altri schemi, differenziandosi solo per quanto riguarda la differente struttura degli ambienti per i singoli tensori della rete.

Nel seguito si suppone che i termini hamiltoniani $h^{[r,r+1]}$ siano caratterizzati da uno spettro non positivo. Questa ipotesi non inficia la generalità della discussione ed è facilmente realizzabile per qualsiasi sistema del tipo (2.9) mediante la sostituzione

$$h^{[r,r+1]} \;\rightarrow\; h^{[r,r+1]} - \lambda_M \mathbb{I},$$

essendo λ_M l'autovalore massimo della matrice $h^{[r,r+1]}$.

Lo stato fondamentale, od eventualmente un sottospazio a basse energie $\mathbb{V}_\mathcal{U} \subset \mathbb{V}_\mathcal{L}$, di un sistema fisico caratterizzato da una Hamiltoniana del tipo (2.9) è convenientemente descritto in termini delle matrici densità ridotte $\rho^{[r,r+i]}$ in modo tale che l'energia $E^{(\mathcal{U})}$, associata al sottospazio $\mathbb{V}_\mathcal{U}$, sia

$$E^{(\mathcal{U})} = \mathrm{tr}\,(H\rho) = \sum_{r\in\mathcal{L}} \mathrm{tr}\left(h^{[r,r+1]}\rho^{[r,r+1]}\right). \tag{2.13}$$

In un MERA l'informazione è codificata nella scelta dei tensori che implementano, ad ogni passo, le rispettive iterazioni del processo di rinormalizzazione. Introdotto l'indice discreto θ che le enumera, la proprietà di dualità della rete di tensori permette di scrivere l'energia $E^{(\mathcal{U})}$ in termini di ciascun reticolo efficace $\mathcal{L}_\theta$ in funzione dei termini hamiltoniani e delle matrici densità ridotte

$$E^{(\mathcal{U})} = \sum_{r\in\mathcal{L}_0} \mathrm{tr}\left(h_0^{[r,r+1]}\rho_0^{[r,r+1]}\right) = \dots = \sum_{r\in\mathcal{L}_\theta} \mathrm{tr}\left(h_\theta^{[r,r+1]}\rho_\theta^{[r,r+1]}\right) = \dots \tag{2.14}$$

Il primo passo di un algoritmo iterativo, per l'ottimizzazione di una rete di tensori, consiste nel determinare come sia possibile scegliere un certo tensore appartenente allo strato θ quando siano fissati tutti gli altri tensori della rete. Oltre al metodo descritto sono stati sviluppati approcci basati sull'utilizzo dell'operatore di evoluzione temporale [13].

Ottimizzazione di un singolo tensore

Consideriamo lo strato θ della rete di tensori, si vuole determinare un algoritmo che permetta di ottimizzarne un disentangler od una isometria. Per semplificare la notazione su indica il tensore considerato con il simbolo generico t, specificandone la tipologia solo quando necessario.

Come visto l'energia puó essere scritta in termini delle Hamiltoniane ristrette e delle matrici densità ridotte, per evidenziare il ruolo delle isometrie che realizzano la ER ricordiamo che le osservabili definite per il reticolo $\mathcal{L}_{\theta+1}$, sono ottenute da quelle definite nel reticolo $\mathcal{L}_\theta$ mediante l'azione di un opportuno operatore di salita, in particolare per le Hamiltoniana ridotte questo si traduce nell'applicazione dell'operatore di salita medio $\overline{\mathcal{A}}$

$$h_{\theta+1}^{[r,r+1]} = \overline{\mathcal{A}}\left(h_\theta^{[r,r+1]}\right), \tag{2.15}$$

che agisce solo sui siti contenuti nel cono casuale passato dei siti r, $r+1$ appartenenti al reticolo $\mathcal{L}_{\theta+1}$. L'energia del sistema diventa allora

$$E^{(\mathcal{U})} = \sum_{r\in\mathcal{L}_{\theta+1}} \mathrm{tr}\left[\overline{\mathcal{A}}\left(h_\theta^{[r,r+1]}\right)\rho_{\theta+1}^{[r,r+1]}\right]. \tag{2.16}$$

Per convenienza si definisce la funzione costo $\mathcal{E}(t)$ del tensore t appartenente allo strato θ, come il contributo all'energia $E^{(\mathcal{U})}$ dipendente esplicitamente da t. Per la struttura dell'operatore di salita, la funzione costo dipende quadraticamente dal tensore t, in particolare devono esistere due famiglie di matrici, che indichiamo con M_s e N_s, tali che

$$\mathcal{E}(t) = \mathrm{tr}\Big(\sum_s t M_s t^\dagger N_s\Big), \tag{2.17}$$

dove l'indice s scorre i siti il cui cono causale contiene il tensore t.

Il problema di scegliere il tensore si riduce a quello di minimizzare la funzione costo nel sottospazio dei tensori che soddisfano il vincolo (2.2). Si tratta quindi di risolvere un problema quadratico vincolato di cui non è noto nessun metodo risolutivo esatto. Per superare questa difficoltà è possibile adottare diverse strategie [3]. In seguito discuteremo un metodo iterativo basato sulla linearizzazione della funzione costo. L'ambiente Υ_t per il tensore t è definito come:

Ambiente Υ_t*:* data una isometria t appartenente ad una rete di tensori, l'ambiente Υ_t è definito come il contributo indipendente da t della funzione costo $\mathcal{E}(t)$, ottenuto considerando indipendenti gli operatori t e $t^\dagger$.

Dall'equazione (2.17) è possibile ricavare la forma esplicita per l'ambiente Υ_t e quindi introdurre la funzione costo linearizzata $\overline{\mathcal{E}}(t)$

$$\Upsilon_t = \sum_s M_s t^\dagger N_s \;\Rightarrow\; \overline{\mathcal{E}}(t) \equiv \mathrm{tr}\left(t\Upsilon_t\right), \tag{2.18}$$

con la convenzione che t e Υ_t siano considerati indipendenti. Il problema di minimizzare la funzione costo linearizzata $\overline{\mathcal{E}}(t)$, con t vincolato, ammette una soluzione nota [7].

Data una matrice $A \in \mathbb{C}^{m\times n}$ (risp. $A \in \mathbb{R}^{m\times n}$), esiste una fattorizzazione di A della forma

$$A = U\Sigma V^\dagger \quad \left(A = U\Sigma V^t\right), \tag{2.19}$$

detta decomposizione ai valori singolari (o *SVD*, dall'inglese Singular Value Decomposition) e tale che

i. la matrice $U \in \mathbb{C}^{m\times m}$ ($U \in \mathbb{R}^{m\times m}$) è unitaria (ortogonale),

ii. la matrice $\Sigma \in \mathbb{C}^{m\times n}$ è una matrice diagonale con elementi reali non negativi sulla diagonale,

iii. la matrice $V \in \mathbb{C}^{n\times n}$ ($V \in \mathbb{R}^{n\times n}$) è unitaria (ortogonale).

La convenzione comunemente utilizzata consiste nell'ordinare gli elementi diagonali di Σ, detti valori singolari di A ed indicati con σ_i per $i = 1, \ldots, \min(m, n)$, in modo non crescente. In questo caso, la matrice diagonale Σ è univocamente determinata da A mentre non lo sono le matrici U e V. Introdotta la matrice I

$$I \in \mathbb{R}^{n\times m} \,:\, I(i,j) = \begin{cases} 0 & \text{se } i \neq j, \\ 1 & \text{altrimenti.} \end{cases}$$

posto $\Upsilon_t = U\Sigma V^\dagger$, la scelta $t = -VIU^\dagger$ minimizza la funzione costo linearizzata

$$\min_t \overline{\mathcal{E}}(t) = \min_t \left\{\mathrm{tr}\left(tU\Sigma V^\dagger\right)\right\} = -\mathrm{tr}\left(I\Sigma\right) = -\sum_i \sigma_i, \tag{2.20}$$

e poiché le matrici V e U sono unitarie è immediato verificare che venga soddisfatto il vincolo (1.14)

$$t^\dagger t = (-VIU^\dagger)^\dagger(-VIU^\dagger) = UIV^\dagger VIU^\dagger = \mathbb{I}. \tag{2.21}$$

Una strategia per ottimizzare il tensore t consiste quindi nell'iterare i seguenti passi n_S volte

S1. Si determina l'ambiente Υ_t con l'ultimo valore di t calcolato.

S2. Si calcolano le matrici U e V che trasformano l'ambiente nella rappresentazione singolare $\Sigma = U^\dagger \Upsilon_t V$.

S3. Il tensore t viene aggiornato mediante la sostituzione $t = -VIU^\dagger$

fino a quando non si ottenga un grado di convergenza soddisfacente. Osserviamo che, in linea di principio, non è garantita convergenza dell'algoritmo, per questo un MERA non puó essere annoverato tra i metodo variazionale. Un'analisi numerica mette peró in evidenza che, almeno nei casi fisicamente rilevanti, si ottengono buoni risultati per n_S dell'ordine di 10.

Si consideri il caso in cui il tensore t sia un disentangler u. Dall'equazione (2.16) risulta che il contributo dovuto a u è contenuto nei tre operatore di salita che contengono u. L'ambiente Υ_u è quindi determinato dalla somma di tre contributi indipendenti, come illustrato nella figura 2.3, corrispondenti

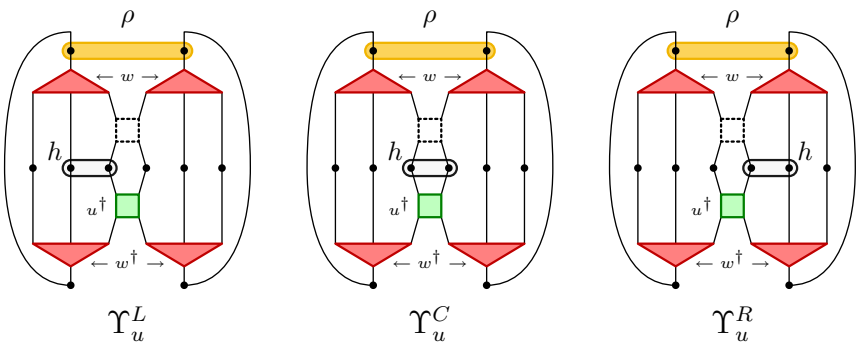


Figura 2.3: l'ambiente Υ_u per il disentangler u è determinato dalla somma dei tre differenti contributi i tre differenti contributi $\Upsilon_u = \Upsilon_u^L + \Upsilon_u^C + \Upsilon_u^R$, le linee tratteggiate rappresentano il tensore rimosso.

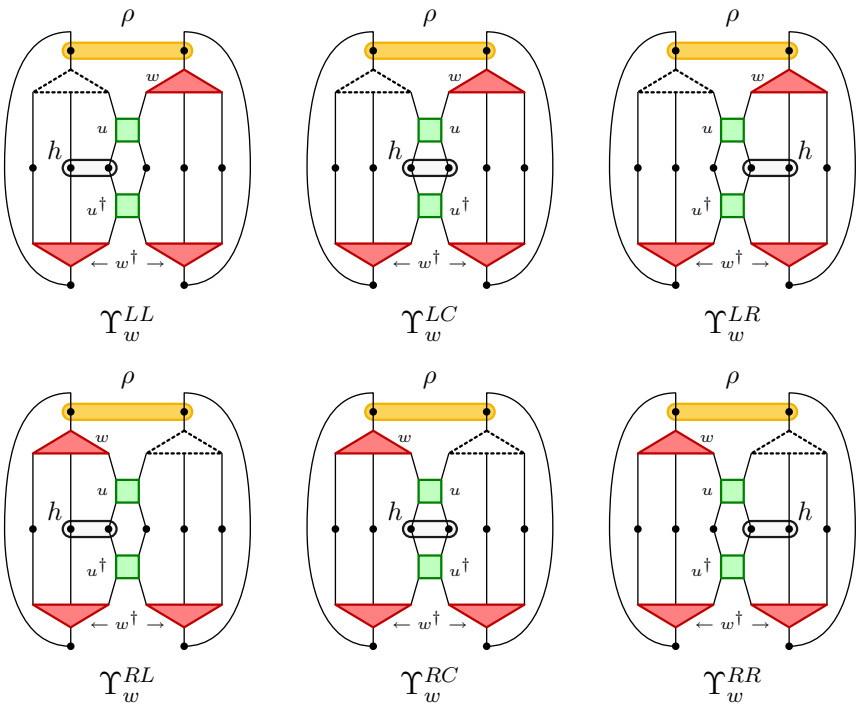


Figura 2.4: nel calcolo dell'ambiente Υ_w per l'isometria w si deve considerare, oltre alla possibile scelta del termine hamiltoniano h, anche quella della posizione relativa dell'isometria rispetto ai disentangler adiacenti. L'ambiente è quindi determinato considerando sei contributi indipendenti $\Upsilon_w = \Upsilon_w^{LL} + \Upsilon_w^{LC} + \Upsilon_w^{LR} + \Upsilon_w^{RL} + \Upsilon_w^{RC} + \Upsilon_w^{RR}$.

ai differenti termini hamiltoniani $h^{[r,r+1]}$ il cui cono causale futuro contiene il disentangler considerato.

Quando il tensore t sia un'isometria w, il relativo ambiente ha una struttura piú complicata in quanto si deve tener conto anche della diversa posizione relativa rispetto ai disentangler adiacenti, in definitiva in tal caso l'ambiente Υ_w è dato dalla somma di sei contributi, figura 2.4.

Invarianza sotto traslazioni

Quando il sistema fisico sia invariante sotto traslazioni è possibile utilizzare una rete di tensori che rispecchi tale proprietà. In tal caso ogni strato è caratterizzato da un solo disentangler e da una sola isometria. Ne segue che, mentre la funzione costo per il disentangler resta una funzione quadratica, quella per l'isometria diviene quartica in w. Ai fini dell'ottimizzazione di singolo tensore si decide comunque di linearizzare la funzione costo trattando come indipendenti i due differenti contributi in w presenti.

Costo computazionale

La notazione grafica, oltre a rappresentare in modo immediato le contrazioni richieste, permette anche di calcolare facilmente il costo computazionale dell'ottimizzazione per singolo tensore. Per ogni iterazione dell'algoritmo di ottmizzazione, è infatti necessario aggiornare l'ambiente Υ_t, come prescritto al passo S1, ma i valori delle Hamiltoniane e delle matrici densità ridotte presenti sono costanti durante tutto il processo. Organizzando opportunamente le contrazioni è quindi possibile determinare l'ambiente Υ_t con un costo computazionele dell'ordine di $\mathcal{O}\left(\chi^8\right)$ determinato contando le contrazioni che riguardano il tensore t, ovvero la parte variabile dell'ambiente. In definitiva ottimizzare un tensore della rete richiede un costo computazione $\mathcal{O}\left(n_S\chi^8\right)$ poichè si devono ripetono n_S volte i passi S1-S3.

2.5 Algoritmo di ottimizzazione

Illustriamo una strategia atta a minimizzare l'energia $E^{(\mathcal{U})}$ del sottospazio $\mathbb{V}_{\mathcal{U}}$ che si deve descrivere, ottimizzando in sequenza i singoli tensori della rete, come descritto nella sezione 2.4. L'idea dell'algoritmo consiste nell'ottimizzare ciascun tensore partendo dallo strato piú basso della rete, ovvero quello indicizzato dal valore $\theta = 0$. Ottimizzati i disentangler e le isometrie appartenenti allo strato θ si aggiornano tutti i termini della Hamiltoniana per lo strato sucessivo $h^{[r,r+1]}_{\theta+1}$ fino a quando $\theta + 1 = \Theta$. Calcolata la Hamiltoniana efficace H_Θ è possibile costruire la relativa matrice densità ρ_Θ. Quindi, mediante l'applicazione degli opportuni operatori di discesa, è possibile aggiornare le matrici densità ridotte di interesse.

Data una rete di tensori, un procedimento iterativo per la sua ottimizzazione che la scorra dal basso verso l'alto, si compone dei seguenti passi:

O1. Per tutti gli strati interni, ovvero quelli enumerati dall'indice θ tale che $\theta = 0, \dots \Theta - 1$:

   i si ottimizzano tutti i disentangler e tutte le isometrie appartenenti allo strato θ,

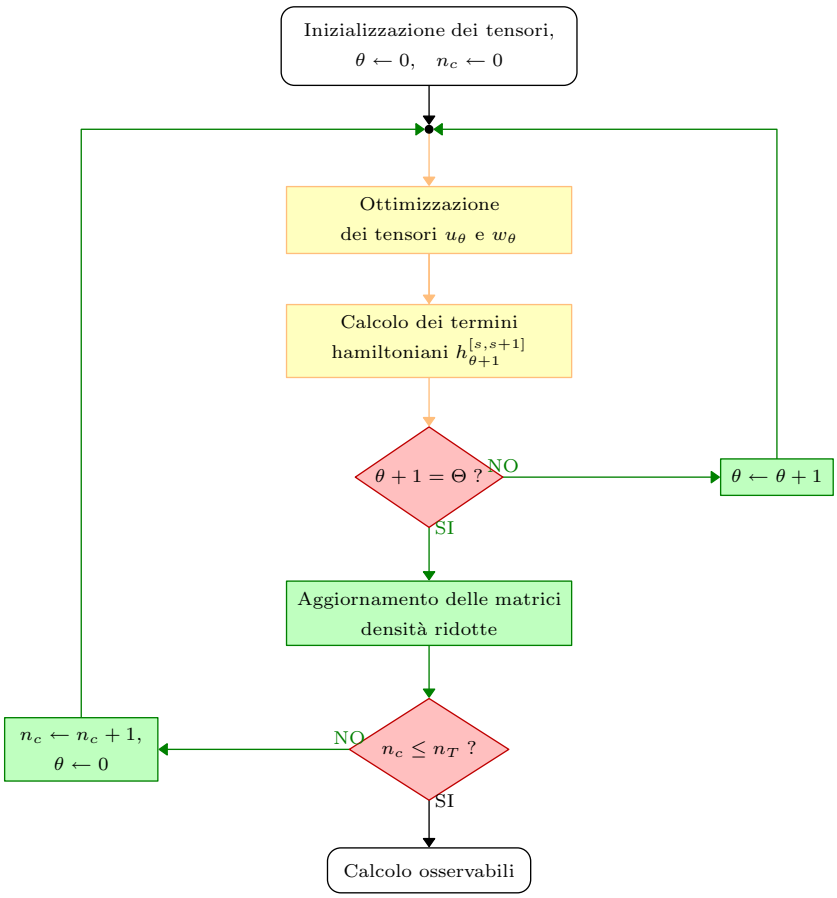


Figura 2.5: diagramma di flusso per un algoritmo di ottimizzazione iterativo che scorra dal basso verso l'alto la rete di tensori.

ii si calcolano i termini hamiltoniani per due siti $h_{\theta+1}^{[r,r+1]}$ applicando gli opportuni operatori di salita ottenuti mediante i tensori aggiornati al passo precedente.

O2. Si aggiorna la matrice densità per lo strato superiore e quindi si determinano le conseguenti matrici densità ridotte per ogni strato inferiore.

Il passo O1.i consiste nell'ottimizzazione progressiva di tutti i tensori appartenenti allo strato θ, mediante l'algoritmo per l'ottimizzazione di singolo tensore illustrato in precedenza. Per ogni tensore considerato dobbiamo costruire l'ambiente Υ_t utilizzando i valori dei termini hamiltoniani e delle matrici densità ridotte, determinati nei passi successivi. Le ottimizzazioni di singolo tensore vengono ripetute un numero n_L di volte, per gli operatori appartenenti allo strato θ. Il costo computazione di questa operazione risulta quindi essere $\mathcal{O}\left(n_L n_S \chi^8 N\right)$, tenendo presente che un MERA non omogeneo è caratterizzato dalla presenza di $\mathcal{O}(N)$ tensori.

Nel passo O1.ii si aggiornano i termini hamiltoniani per lo strato successivo utilizzando i disentangler e le isometrie risultate da O1.i. Il costo computazionale di questa operazione è di ordine $\mathcal{O}\left(\chi^8\right)$ per ciascun termine, corrispondente all'applicazione dell'opportuno operatore di salita $\mathcal{A}$.

Si prosegue fino a quando non sia spaziata tutta la rete di tensori. Ottenuta la Hamiltoniana efficace H_Θ, è possibile determinare la matrice densità efficace ρ_Θ utilizzando metodi di diagonalizzazione esatta. Quindi si aggiornano le matrici densità ridotte per tutti gli strati sottostanti con un costo computazionale dell'ordine $\mathcal{O}\left(\chi^8 N\right)$.

Il processo viene ripetuto un numero totale n_T di volte fino al raggiungimento del grado di accuratezza desiderato. Il costo computazionale totale per ottimizzare un MERA risulta quindi essere dell'ordine $\mathcal{O}\left(n_T n_L n_S \chi^8 N\right)$.

Dall'analisi della convergenza dei risultati prodotti dall'algoritmo, si evince come risulti opportuno utilizzare valori di n_L e n_S relativamente piccoli, solitamente dell'ordine di 5: non è conveniente ottimizzare tensori utilizzando funzioni costo che sono destinate a variare nelle iterazioni successive.

Ottimizzazione in presenza di simmetrie

UN sistema invariante sotto traslazioni puó essere descritto da una MERA invariante sotto traslazioni, con la conseguente sensibile riduzione del numero dei parametri da considerarsi, traendone vantaggio anche l'algoritmo di ottimizzazione. Il passo i1 si riduce infatti all'ottimizzazione di una singola coppia di tensori, un disentangler ed una isometria.

Analogamente abbiamo un solo termine hamiltoniano ed una sola matrice densità ridotta per ogni valore di θ, il costo computazionale associato al passo O2 si riduce a $\mathcal{O}\left(\chi^8 \log N\right) \approx \mathcal{O}\left(\chi^8 \Theta\right)$ corrispondente alla ripetuta applicazione dell'operatore di discesa medio $\overline{\mathcal{D}}$. Segue quindi che il costo computazionale complessivo per ottimizzare un MERA invariante sotto traslazioni si riduce a $\mathcal{O}\left(n_T n_L n_S \chi^8 \log N\right)$.

Il costo computazionale per un sistema descritto da un MERA invariante di scala, subisce un'ulteriore riduzione in quanto i tensori indipendenti si riducono ad essere $\bar{\theta} < \theta$. Procedendo come in precedenza si ottiene un costo computazionale $\mathcal{O}\left(n_T n_L n_S \chi^8 \log \bar{N}\right)$, dove si è convenientemente introdotto il parametro $\bar{N} < N$. In definitiva un MERA invariante sotto trasformazioni di scala ammette un costo computazionale per la sua ottimizzazione indipendente dalle dimensioni del reticolo fisico descritto [11].

2.6 Cenni sull'implementazione

Si è implementato L'algoritmo per un MERA invariante sotto traslazioni utilizzando il linguaggio C++. La tecnica dei template di espressioni ha permesso di sviluppare una interfaccia naturale per le contrazioni tra tensori, descritta in dettaglio nell'appendice A.

La flessibilità del formalismo sviluppato ha permesso di effettuare, con relativa facilità, ottimizzazioni che sarebbe state di complicata implementazione utilizzando altri linguaggi di programmazione, come il fortran o il C. In particolare, una volta implementato l'algoritmo, è stato possibile riorganizzare le contrazioni in modo efficente.

Raffinamento dell'algoritmo di ottimizzazione

In precedenza si è stimato il costo computazionale necessario ad ottimizzare un tensore della rete, come $\mathcal{O}\left(n_S\chi^8\right)$. Tale stima era ottenuta considerando le contrazioni che riguardano i termini dipendenti dal disentangler o dall'isometria nel relativo ambiente. È possibile migliorare questo risultato organizzando opportunamente le contrazioni.

Per semplificare la notazione si considera esplicitamente il contributo legato al termine Υ_u^C. La figura 2.3 rappresenta le contrazioni

$$(\Upsilon_u^C)_{rs}^{tv} = w^{*i}{}_{abc} w^{*j}{}_{def} u^{*cd}{}_{pq} h_\theta{}^{pq}_{rs} w_h^{abt} w_k^{vef} \rho_{\theta+1}{}_{ij}^{hk}, \tag{2.22}$$

i termini che non contraggono direttamente il tensore u, possono essere isolati e devono essere calcolati solo una volta durante l'ottimizzazione. Il conseguente tensore Y

$$Y_{cf}^{tv} = w^{*i}{}_{abc} w^{*j}{}_{def} w_h^{abt} w_k^{vef} \rho_{\theta+1}{}_{ij}^{hk}, \tag{2.23}$$

organizzando opportunamente le contrazioni, viene calcolato con un costo computazionale $\mathcal{O}\left(\chi^6\right)$

$$\begin{aligned} W_{ch}^{it} &= w^{*i}{}_{abc} w_h^{abt} && (\mathcal{O}\left(\chi^6\right)) \\ V_{dk}^{jv} &= w^{*j}{}_{def} w_k^{vef} && (\mathcal{O}\left(\chi^6\right)) \\ \tilde{\rho}_{cj}^{tk} &= W_{ch}^{it} \rho_{\theta+1}{}_{ij}^{hk} && (\mathcal{O}\left(\chi^6\right)) \\ Y_{cf}^{tv} &= \tilde{\rho}_{cj}^{tk} V_{dk}^{jv} && (\mathcal{O}\left(\chi^6\right)) \end{aligned}$$

In funzione del tensore Y, l'ambiente Υ_u^C diventa

$$(\Upsilon_u^C)_{rs}^{tv} = Y_{cf}^{tv} h_\theta{}^{pq}_{rs} u^{*cd}{}_{pq}, \tag{2.24}$$

procedendo come per il tensore Y, Υ_u^C viene calcolato con un costo computazionale $\mathcal{O}\left(\chi^6\right)$. Il procedimento illustrato, puó essere ripetuto anche per gli ambienti Υ_u^L e Υ_u^R, in questi casi si guadagna solo un fattore χ mentre la parte costante necessita di un costo computazionale di ordine $\mathcal{O}\left(\chi^8\right)$.

In definitiva si è ottenuta una implementazione dall'algoritmo di ottimizzazione per il disentangler, caratterizzata da un costo computazionale $\mathcal{O}\left(\chi^8 + n_S\chi^7\right)$.

La natura più complicata del problema che riguarda l'isometria w non permette di estendere i risultati a questo caso quando si considerino sistemi invarianti sotto traslazioni.

Capitolo 3

Modello di Ising

Applichiamo l'algoritmo allo studio delle proprietà dello spettro a basse energie per il modello di Ising in campo magnetico esterno trasverso a temperatura zero. Consideriamo l'Hamiltoniana

$$H = \sum_s \left(\lambda \sigma_z^{[s]} + \sigma_x^{[s]} \sigma_x^{[s+1]} \right), \tag{3.1}$$

dove $\sigma_a^{[s]}$ è la matrice di Pauli che agisce sul sito s:

$$\sigma_a^{[s]} = \mathbb{I} \otimes \mathbb{I} \cdots \otimes \sigma_a \otimes \cdots \mathbb{I} \otimes \mathbb{I}$$

e la somma è estesa a gli N siti del reticolo $\mathcal{L}$. Il parametro λ rappresenta l'intensità del campo magnetico esterno applicato lungo l'asse z. Una transizione di fase si realizza per il valore critico $\lambda_c = 1$. In una dimensione, sono note soluzioni analitiche implicite del modello [14], che puó quindi essere utilizzato per verificare la correttezza dell'algoritmo sviluppato.

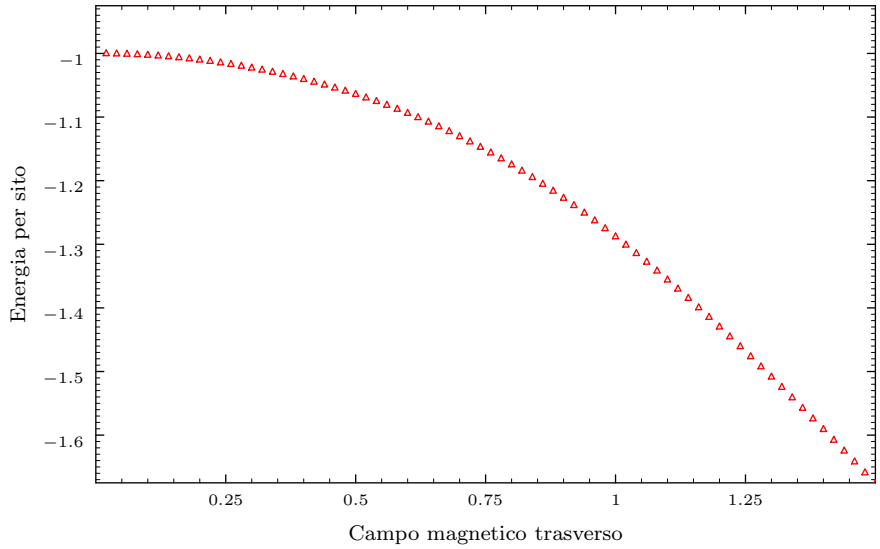
Assumendo condizioni al contorno periodiche, è possibile utilizzare un MERA invariante sotto traslazioni con il vantaggio sul costo computazionale che ne consegue.

3.1 Modello a piccole dimensioni

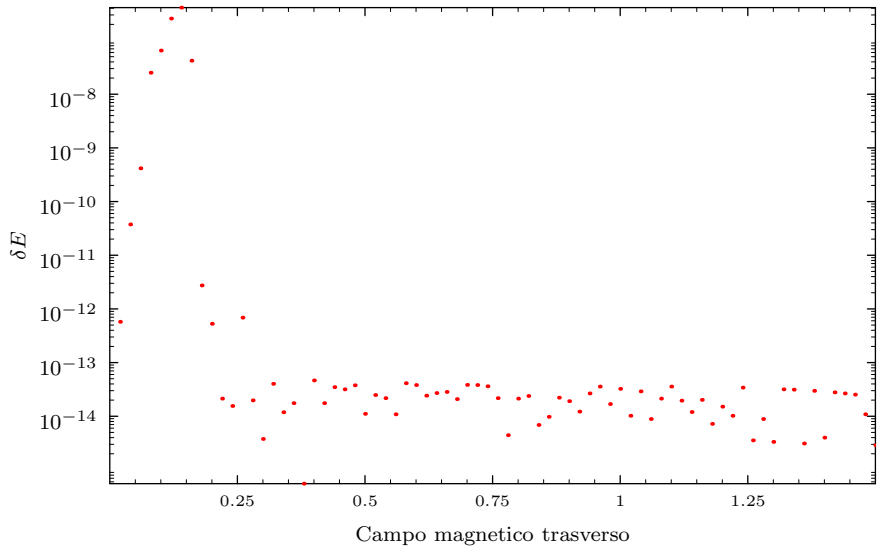
Sistemi di piccole dimensioni sono interessanti perchè consentono di confrontare i risultati ottenuti con le soluzioni fornite da metodi di diagonalizzazione esatti, anche quando non sia nota una soluzione analitica e permettono di evidenziare alcune proprietà dell'algoritmo, utili quando si vadano a considerare sistemi di dimensioni maggiori.

Consideriamo un reticolo composto da sei siti, descritto quindi da una rete di tensori organizzata in due strati. Quando il parametro di rifinitura χ sia uguale a 8, la ER si riduce ad una semplice trasformazione di coordinate e quindi riproduce esattamente tutte le proprietà del sistema.

Poniamo quindi $\chi = 4$, abbiamo utilizzato l'algoritmo per determinare l'energia dello stato fondamentale come riportato in figura 3.1. Si osserva che l'algoritmo produce risultati numericamente esatti per $\lambda > 0.25$, nella regione di piccolo campo magnetico la capacità risolutiva si riduce con un errore massimo dell'ordine $\mathcal{O}\left(10^{-6}\right)$ comunque soddisfacente. L'algoritmo



(a)



(b)

Figura 3.1: energia (a) e relativo errore (b) per un sistema composto da $N = 6$ siti corrispondente ad un MERA costituito da due strati. I risultati sono ottenuti considerando un parametro di rifinitura $\chi = 4$, simulazioni effettuate per valori maggiori riproducono esattamente le proprietà del sistema, in particolare per $\chi = 8$ la Hamiltoniana H_Θ corrisponde, a meno di una trasformazione di similitudine, a quella esatta del modello.

riproduce gli effetti del modello a dimensioni finite, come confermato dallo studio della magnetizzazione trasversa media $\langle \sigma_z \rangle$, figura 3.2. Anche in questo caso l'andamento dell'errore mostra una maggiore difficoltà in un intorno della regione $\lambda \approx 0.15$, mentre si ottengono risultati numericamente esatti per $\lambda > 0.25$.

Applicare il MERA ad un sistema composto da sei siti equivale a considerare una rete di tensori caratterizzata da due strati e quindi, nel caso invariante sotto traslazioni, una sola coppia (u, w). Il disentangler viene scelto per rimuovere l'intrecciamento tra i due blocchi in cui è partizionato il reticolo, quale che sia la scelta del parametro χ, nel caso considerato è un tensore di dimensione $\chi_0 = 2$ in quanto, per costruzione si applica a due siti appartenenti al reticolo fisico. L'isometria w determina il un sito efficace per il reticolo $\mathcal{L}_1$, la sua efficacia dipende dall'azione del disentangler: il maggiore errore riscontrato nella regione $\lambda \approx 0.15$ suggerisce che in queste condizioni il disentangler non sia in grado si selezionare i gradi di libertà fisicamente irrilevanti e che quindi sia necessario allargare la dimensione del sito efficace.

Questa osservazione risulterà particolarmente utile quando si considereranno sistemi di dimensioni maggiori.

3.2 Limite termodinamico

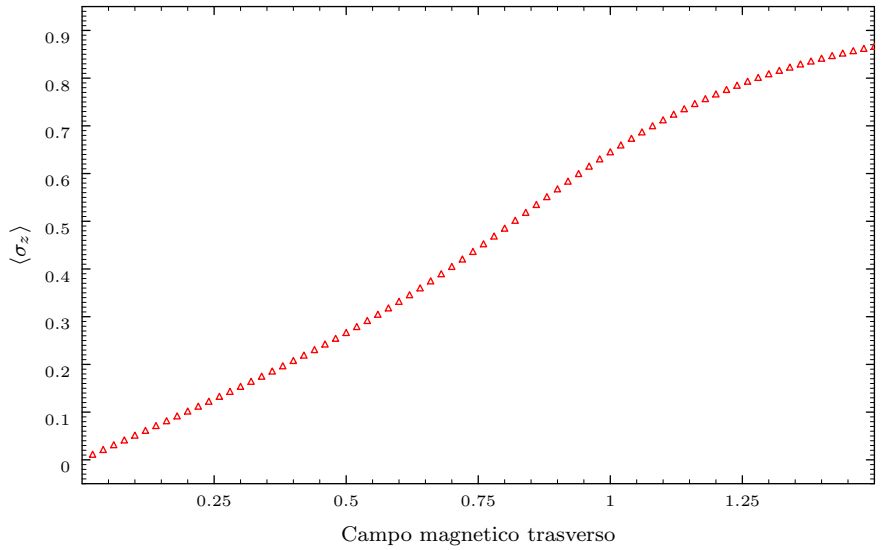
Un MERA caratterizzato da $\Theta = 4$ descrive un reticolo composto da $N = 162$ siti, i risultati ottenuti in questo caso non dipendono significativamente delle dimensioni del sistema, ovvero siamo al limite termodinamico.

Si è quindi calcolata l'energia dello stato fondamentale del sistema e la magnetizzazione trasversa al variare del campo magnetico trasverso. I risultati sono stati confrontati con le soluzioni analitiche esatte e l'errore ottenuto è riportato in figura 3.3. Per minimizzare gli errori commessi si sono tenute presenti due proprietà, comunque evidenziate da una analisi numerica preliminare, e quindi generalizzabili anche per altri modelli.

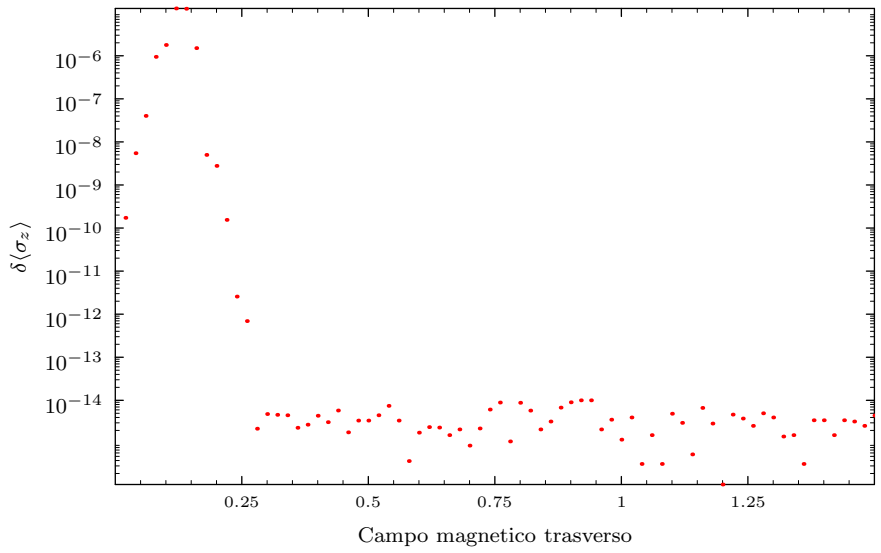
Per λ piccolo, nel modello di Ising, la differenza di energia tra lo stato fondamentale ed il primo stato eccitato tende a zero, inducendo oscillazioni nella convergenza dell'algoritmo di ottimizzazione: l'algoritmo risulta incapace di distinguere esattamente lo stato fondamentale. Questa problematica puó essere risolta decidendo di descrivere un sottospazio piú ampio $\mathbb{V}_{\mathcal{U}}$ anziché ridurci al solo stato fondamentale. Cosí facendo si sono eliminate tutte le fluttuazioni presentatesi durante la fase di ottimizzazione della rete di tensori.

Durante le simulazioni effettuate con $\chi = 4$ si è inizialmente osservata una difficoltà nell'ottenere buoni risultati per $\lambda \approx 0.25$, non giustificata dell'andamento dell'errore che si delineava nelle regioni adiacenti. In tal caso l'esperienza fatta per sistemi di piccole dimensioni ha suggerito di aumentare la dimensione di sito solo per il reticolo $\mathcal{L}_1$. Con questo accorgimento non si modifica sostanzialmente il costo computazionale per la simulazione, ma si sono migliorati i risultati ottenuti, esposti in figura 3.3.

Al valore $lambda = 1$ corrisponde il punto critico per il modello di Ising, in questo caso si ottengono le stime peggiori per l'energia dello stato fondamentale. Anche nel caso peggiore, le simulazioni effettuate con $\chi = 4$, si ottiene comunque un risultato confortante essendo $\delta E \approx 10^{-5}$. Aumentando il parametro χ aumenta l'accuratezza: per $\chi = 8$ l'errore commesso è dell'ordine di 10^{-6}.

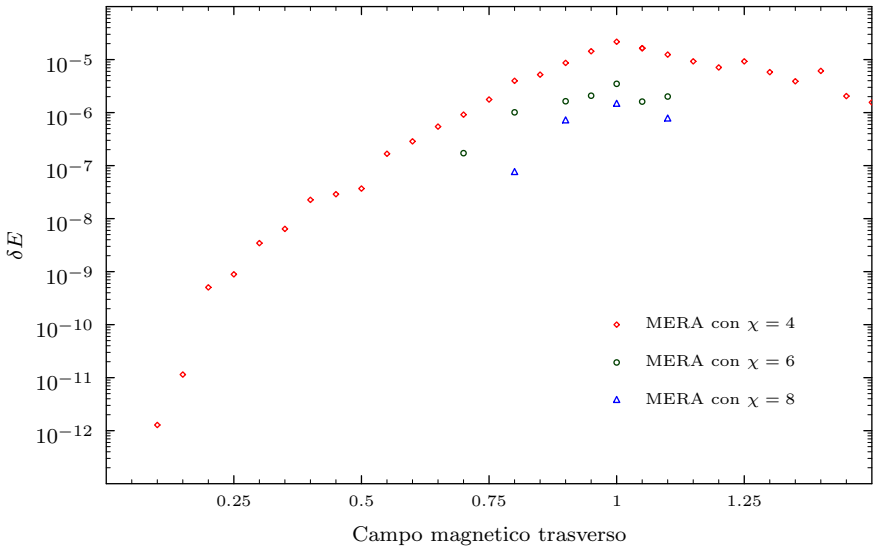


(a)

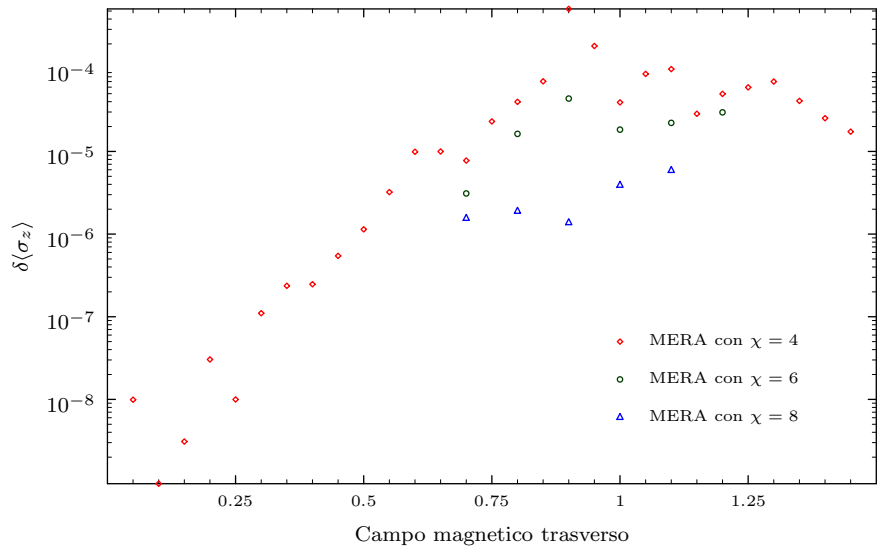


(b)

Figura 3.2: polarizzazione media (a) e relativo errore (b) per un sistema composto da $N = 6$ siti. I risultati sono ottenuti considerando un parametro di rifinitura $\chi = 4$, e concordano con .



(a)



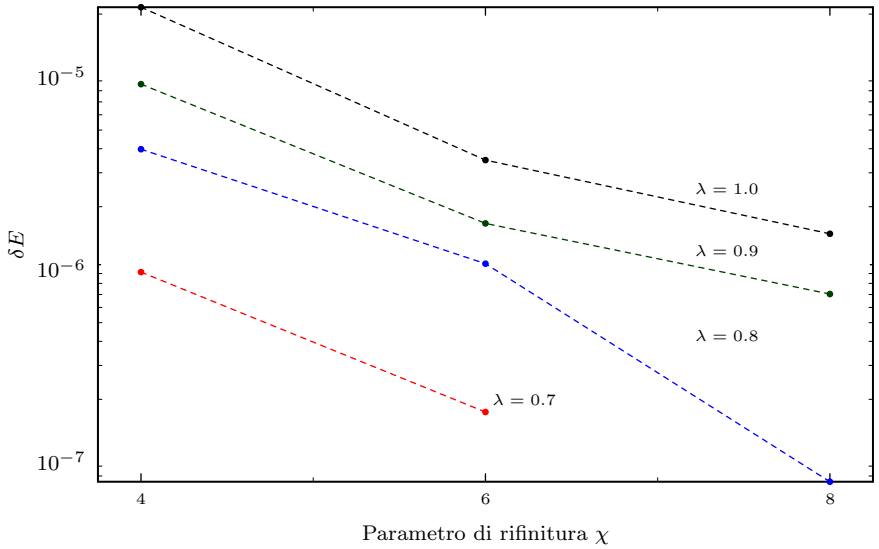
(b)

Figura 3.3: errore per l'energia (a) e per la magnetizzazione trasversa (b) per un sistema composto da $N = 162$ siti corrispondente ad un MERA costituito da cinque strati. Si sono effettuate simulazioni considerando un valore del parametro di rifinitura uguale a $\chi = 4$, $\chi = 6$ e $\chi = 8$. I risultati ottenuti sono confrontati con il limite termodinamico.

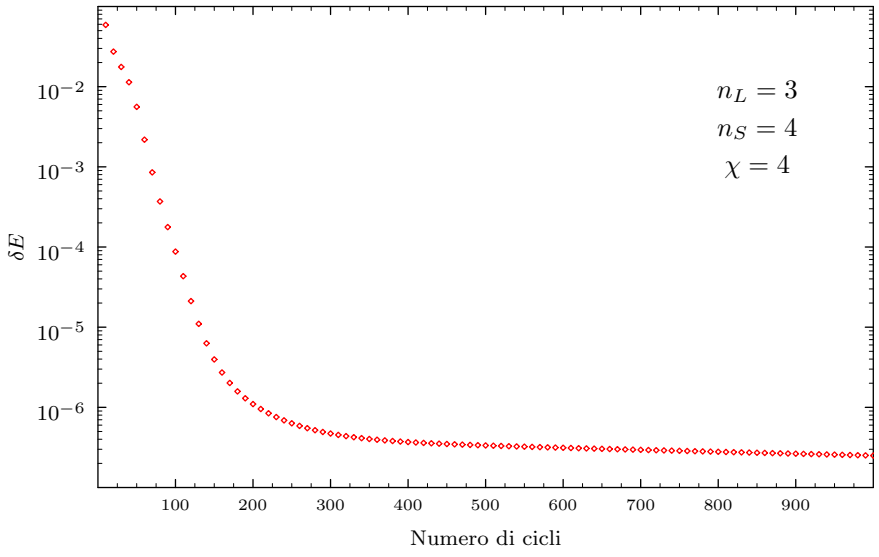
La relativa difficoltà dell'algoritmo nel trattare sistemi critici rappresenta una conferma alla congettura per cui il punto critico di una catena si spin quantistici, corrisponde alla configurazione in cui sia massimizzato l'entanglement del sistema [10].

Durante la discussione dell'algoritmo, si è spesso espresso il costo computazionale in funzione del parametro di rifinitura χ, mostrando come questo sia solitamente polinomiale. L'errore commesso invece diminuisce esponenzialmente con χ come mostrato dalla figura 3.4a, questa proprietà giustifica a posteriori il nome attribuito al parametro di rifinitura.

Un altro parametro che influenza il costo computazionale, necessario ad ottimizzare una rete di tensori, è il numero totale di cicli effettuati n_T. In figura 3.4b si è riportato l'andamento dell'errore in funzione del numero di iterazioni effettuate. Si osserva un andamento almeno esponenziale per i primi 200 cicli, dopo di che l'errore si stabilizza tendendo al valore asintotico.



(a)



(b)

Figura 3.4: andamento dell'errore in funzione del parametro χ per diversi valori del campo magnetico esterno (a) ed in funzione del numero di cicli di ottimizzazione eseguiti (b). In questo caso la simulazione è stata effettuata per $\chi = 4$.

Conclusione

In questa tesi si è studiato un algoritmo per sistemi quantistici unidimensionali, MERA, basato sulla rinormalizzazione di entanglement. In questi metodi, lo spazio di Hilbert è soggetto a trasformazioni locali atte a ridurre l'intrecciamento tra i blocchi del sistema, prima che vengano selezionati gli stati fisicamente rilevanti.

Durante lo sviluppo dell'algoritmo è emersa la necessità di implementare in modo efficiente le contrazioni che realizzano le trasformazioni del gruppo di rinormalizzazione. Il linguaggio C++ ha messo a disposizione i mezzi necessari. Utilizzando la tecnica dei template di espressioni, si è realizzata un'interfaccia per tensori simile a quella comunemente utilizzata in fisica.

Si è quindi testato l'algoritmo prendendo come riferimento il modello di Ising, per il quale sono note soluzioni analitiche.

Un'analisi numerica dei risultati ha mostrato come il MERA sia un metodo che descrive in modo conveniente sistemi invarianti sotto traslazioni e sistemi critici. In questo caso il costo computazionale è indipendente dalle dimensioni del sistema permettendo di considerare il limite termodinamico.

Appendice A

Tensori

I tensori sono oggetti matematici che includono ed estendono le nozioni di vettori e matrici dell'algebra lineare e che trovano una naturale implementazione numerica in termini di array multidimensionali. Tale implementazione presenta peró alcune limitazioni, in particolare non fornisce un'interfaccia naturale per manipolare gli oggetti definiti che ricordi quella comunemente utilizza in fisica.

Per fissare le idee consideriamo una coppia di spazi vettoriale $\mathbb{V}$, $\mathbb{W}$ ed una trasformazione lineare T che mappi $\mathbb{V}$ in $\mathbb{W}$:

$$\mathbb{V} \ni v \longrightarrow T(v) = w \in \mathbb{W} \tag{A.1}$$

Solitamente si rappresentano gli spazi vettoriali introducendo un sistema di coordinate, sia quindi $\mathcal{B}^{(\mathbb{V})} = \{e^1, e^2, \ldots, e^n\}$ una base di $\mathbb{V}$ e $\mathcal{B}^{(\mathbb{W})} = \{f^1, f^2, \ldots, f^m\}$ una di $\mathbb{W}$ in modo tale i vettori v e w possano essere scritti come combinazioni lineari degli elementi delle basi:

$$v = v_1 e^1 + v_2 e^2 + \cdots + v_n e^n \quad \text{e} \quad w = w_1 f^1 + w_2 f^2 + \cdots + w_m f^m,$$

analogamente l'operatore T puó essere espresso in termini delle basi, in definitiva l'applicazione della trasformazione lineare ad uno spazio vettoriale viene rappresentata dell'usuale prodotto tra una matrice ed un vettore:

$$\begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} T_{11} & \cdots & T_{1n} \\ \vdots & \ddots & \vdots \\ T_{m1} & \cdots & T_{mn} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}. \tag{A.2}$$

L'equazione (A.2) riscritta per la componente i-esima del vettore w assume una forma piú semplice

$$w_i = \sum_{j=1}^{n} T_{ij} v_j \quad \text{dove } i \in \{1, \ldots, m\},$$

ed introducendo la convenzione di sommare gli indici ripetuti se ne determina una rappresentazione particolarmente compatta *(notazione di Einstein)*:

$$w_i = T_{ij} v_j, \tag{A.3}$$

la cui utilità risulta evidente considerando espressioni piú complicate che contengano tensori di rango superiore come ad esempio la definizione del tensore

di Riemann in relatività generale:

$$R^{\rho}{}_{\sigma\mu\nu} = \partial_{\mu}\Gamma^{\rho}{}_{\nu\sigma} - \partial_{\nu}\Gamma^{\rho}{}_{\mu\sigma} + \Gamma^{\rho}{}_{\mu\eta}\Gamma^{\eta}{}_{\nu\sigma} - \Gamma^{\rho}{}_{\nu\eta}\Gamma^{\eta}{}_{\mu\sigma}.$$

Di fronte ad espressioni complicate un'implementazione classica dei tensori risulta non ottimale e per efficienza e per comoditá dell'interfaccia realizzata, il C++ offre comunque le risorse necessarie per risolvere questo problema in modo elegante mantenendo al contempo buone prestazioni.

A.1 Metaprogrammazione template

I linguaggi orientati ad oggetti offrono internamente la possibilità di creare astrazioni necessarie nello sviluppo di librerie specie in ambito scientifico. In particolare le classi e il sovraccaricamento degli operatori permettono inoltre di mimare piuttosto fedelmente la sintassi delle operazioni matematiche che si intende implementare.

Solitamente tali implementazioni sono associate ad una *penalità per astrazione* dovuta alla difficoltà del compilatore di generare codice ottimizzato: l'astrazione offusca al compilatore la struttura semantica degli oggetti che elabora.

Il meccanismo dei template aggira, almeno parzialmente, tale limitazione permettendo di creare librerie che ammettono una sintassi astratta mantenendo tuttavia prestazioni spesso paragonabili a quelle di codice ottimizzato manualmente.

I template, nati per supportare la programmazione generica nel linguaggio C++, sono quindi di grande utilità in quanto permettono di scrivere codice altamente riutilizzabile mantenendo tuttavia le peculiarità della programmazione orientata agli oggetti. Accidentalmente questo meccanismo formisce altre importanti funzionalità come la possibilità di effettuare calcoli statici o di generare codice. Gran parte della libreria standard C++ è implementata utilizzando template.

Programmazione generica

I template sono il mezzo attraverso il quale il linguaggio C++ supporta la programmazione generica secondo un paradigma che è possibile sintetizzare nell'espressione "riutilizzo via parametrizzazione". Il template di una classe o di una funzione definisce una famiglia di classi e di funzioni rispettivamente.

Il codice generico definito attraverso il meccanismo dei template non viene elaborato fino a quando non se ne richieda una istanza esplicitando i valori dei parametri che lo caratterizzano, questo implica che i parametri del template debbano essere staticamente vincolati, ovvero fissati a tempo di compilazione.

Ad esempio possiamo definire una classe che fornisca una implementazione di un vettore, utilizzando il metodo dei template il prototipo realizzato non è vincolato ad un tipo specifico, ma costituisce un contenitore generico:

```
template<typename T, int N>
class Vector {
  T _data [N];

5    ...
```

```
};

// Utilizzo della classe Vector
Vector<float, 4> x;
Vector<float, 5> y;
```

Questo semplice esempio di modello di classe illustra il ruolo che possono avere i parametri, nel caso specifico `T` ed `N`:

- `T` è un *parametro tipo*, determina il tipo di dato contenuto nella classe `Vector`, ed è associato alle parole chiave `typename` o `class` indistintamente,
- `N` è un *parametro non-tipo*, specifica la dimensione del vettore, e puó essere un valore qualsiasi del tipo fondamentale presente nella dichiarazione del template.

Nella linea 10 si dichiara un vettore di `float`, `x`, contenente quattro elementi, mentre in quella seguente se ne definisce uno di cinque elementi, `y`.

Osserviamo che un ipotetico controllo di tipo tra `x` e `y` è destinato a fallire: differenti parametri definiscono classi diverse. Questa proprietà ha importanti ripercussioni che verranno analizzate in seguito.

Calcoli statici e generazione di codice

Come visto il meccanismo di parametrizzazione implementato in C++ permette di utilizzare come parametri tipi o valori, inoltre è ammessa la definizione di specializzazioni parziali o complete del template. Utilizzando questa caratteristica è possibile forzare il compilatore ad eseguire calcoli statici.

Questa proprietà è in un certo senso accidentale ed è stata scoperta per la prima volta da Erwin Unruh [16], il quale ha scritto un programma che calcola una lista di numeri primi e la restituisce tra i messaggi di errore prodotti dal compilatore:

```
error: initializing argument 1 of 'D<i>::D(void*) [with int i=13]'
error: initializing argument 1 of 'D<i>::D(void*) [with int i=11]'
error: initializing argument 1 of 'D<i>::D(void*) [with int i=7]'
error: initializing argument 1 of 'D<i>::D(void*) [with int i=5]'
error: initializing argument 1 of 'D<i>::D(void*) [with int i=3]'
error: initializing argument 1 of 'D<i>::D(void*) [with int i=2]'
```

Per illustrare il funzionamento di questo meccanismo, consideriamo un caso piú semplice rispetto a quello ideato da Unruh. Vogliamo costruire una classe che calcoli il fattoriale per un dato intero N mediante template valutati a tempo di compilazione. Iniziamo definendo una modello per una classe parametrizzata dall'intero N che contiene un solo membro costante `result`. Dalla definizione della classe (listato A.1) si evince che richiedere il membro `result` per $N > 0$ impone al compilatore di creare istanze della classe per $N-1, N-2, \ldots$ fino a quando non subentra la specializzazione della classe per $N = 0$ (Linea 8). La natura costante degli oggetti considerati impone che vengano istanziati a tempo di compilazione, stiamo quindi forzando il compilatore a comportarsi di fatto come un interprete.

La capacitá di eseguire calcoli statici è sorprendente ma rispecchia una proprietà strutturale del C++, osserviamo infatti che la possibilità di definire una specializzazione del codice template costituisce un construtto condizionale

```
#include <stdio.h>

template<int N>
struct factorial {
  static const int result = N * factorial<N-1>::result;
};

template<>
struct factorial<0> {
  static const int result = 1;
};

// z deve essere calcolato staticamente
const int z = factorial<4>::result;
```

listato A.1: Codice per il template di classe `factorial<int N>`, poiché in questo esempio semplificato vogliamo che ogni menbro della classe sia pubblico, utilizziamo la parola chiave `struct` al posto della usuale `class`.

statico in quanto il compilatore deve essere in grado di effettuare una scelta tra le diverse possibili. Inoltre esiste un costrutto statico per cicli poiché il linguaggio ammette chiamate ricorsive anche nel codice template (ad esempio è possibile definire funzioni template che chiamano se stesse).

Per questo il metodo dei template, detto comunemente metaprogrammazione template, costituisce un sottolinguaggio Turing completo[1] eseguito a tempo di compilazione del linguaggio C++. Almeno teoricamente non esistono limiti a quello che si può implementare con tale sottolinguaggio, anche se solitamente esistono restrizioni legate al compilatore utilizzato.

Nello stesso spirito è possibile forzare il compilatore a generare codice (listato A.2), si vuole rimarcare che in questo caso sono valutate a tempo di compilazione le istanze del template `dot` mentre gli argomenti `x`, `y` sono indeterminati fino al momento dell'esecuzione, quando viene quindi valutata la specializzazione creata della funzione generica. Il template ha una natura ibrida, parliamo in questo caso di valutazione parziale [6, 17].

Template di espressioni

Passare un'espressione ad una funzione è un evento comune. In C, le espressioni sono di solito implementate utilizzando un puntatore a una funzione di callback che le realizzano. Ad esempio, la libreria standard C contiene routine come `qsort()`, `lsearch()`, e `bsearch()` che accettano come argomento un `int (*cmp)(void*, void*)` cioè un puntatore ad una funzione definita dall'utente che determina come effettuare il confronto tra due elementi generici. Se l'espressione considerata è relativamente semplice, questo metodo presenta prestazioni intrinsecamente limitate dovute al meccanismo di chiamata che deve essere forzatamente dinamico: in questo caso il sistema "spreca" una quantità di risorse a richiamare le funzioni di callback paragonabile a quella necessaria ad eseguire la routine richiesta.

[1] Un modello di calcolo è detto Turing completo se ha lo stesso potere computazionale di una Macchina di Turing Universale, i linguaggi di programmazione sono Turing completi se contengono almeno un costrutto per cicli ed uno condizionale.

```
template<int I>
inline float dot(float* a, float* b) {
  return dot<I-1>(a,b) + a[I]*b[I];
}

template<>
inline float dot<0>(float* a, float* b) {
  return a[0]*b[0];
}

// Example use:
float x[3], y[3];
float z = dot<2>(x,y);
```

listato A.2: Mediante chiamate ricorsive, effettuate staticamente dal compilatore, la chiamata a dot equivale al codice `a[0]*b[0] + a[1]*b[1] + a[2]*b[2]`.

I template di espressioni [18] offrono una soluzione a questo problema in quanto le funzioni vengono passate come parametro di un template e quindi incorporate nel codice a tempo di compilazione. Nel contesto della metaprogrammazione, i template di espressioni sono un esempio di valutazione parziale: le operazioni matematiche contenute nelle espressioni sono trattate come oggetti statici, mentre la loro valutazione avviene dinamicamente.

A.2 Implementazione di una notazione naturale

Introdotti gli strumenti necessari, descriviamo come si è potuto realizzare una interfaccia naturale per tensori. Per semplificare la notazione ci limitiamo a tensori di rango uno, tensori di rango superiore sono facili generalizzazioni del caso trattato.

Classi base

L'idea alla base dell'implementazione consiste nel separare la struttura dati, che deve andare a contenere i valori numerici rappresentanti il tensore, e le espressioni astratte che permettono di manipolare l'oggetto considerato. La struttura dati è implementata attraverso un contenitore generico (listato A.3), ovviamente volendo trattare questo oggetto come un tensore si è fornita un'interfaccia per accedere all'elemento i-esimo del tensore.

Si vuole supportare una notazione del tipo $w(i) = v(i)$ dove l'indice i sia sommato implicitamente:

$$w(i) = v(i) \longrightarrow \forall\, i = 1, \ldots, n \text{ do } w(i) = v(i),$$

per ottenere questo dobbiamo introdurre due classi, una che permetta di distinguere l'indice i, detta `Index`, ed una che contenga le espressione relative a quantità che si riducono ad un tensore di rango uno, detta `T1Expr`.

Il template `Index` è molto semplice e contiene un solo parametro `char i`:

```
template<char i>
class Index {};
```

```
template<class T>
class T1 {
  T* _e;
  size_t _dim1;

  void init() {
    _e = new T[_dim1];
  }

  void clear() {
    if (_e != 0) {
      delete [] _e;
    }
  }

  public:
  T1 (const size_t d) : _e (0), _dim1 (d) { init(); }
  ~T1 () { clear(); }

  inline T& operator() (const size_t i) {
    return *(_e + i);
  }

  inline size_t dim1() const { return _dim1; }
};
```

listato A.3: Schematizzazione del template `T1<typename T>`

le sue istanze sono utilizzate per specificare al compilatore quale indice, o indici per tensori di rango superiore, considerare attraverso un controllo sui tipi statico, ricordiamo infatti che per parametri diversi del template si vengono a creare istanze di classi differenti.

Il template `T1Expr` è destinato a contenere tutte le espressioni che si riducono a tensori di rango uno come le seguenti relazioni matematiche

$$a_i + b_i, \quad w_j w_j^* v_i, \quad A_{ij} w_j, \quad \ldots$$

per realizzare questo utilizziamo tre parametri: `class A` che indica che tipo di oggetto l'espressione andrà a contenere, `typename T` che rappresenta il tipo fondamentale contenuto dalla precedente classe e `char i` che indica l'indice dell'espressione:

```
template<class A, typename T, char i>
class T1Expr {
  A _ref;
  size_t _dim1;

public:
  T1Expr (A& a) : _ref (a), _dim1 (a.dim1()) {}

  T operator() (const size_t n) const {
    return _ref(n);
  }

  size_t dim1() const { return _dim1; }
};
```

```
template<class T, char i>
class T1Expr<T1<T>, T, i> {
  T1<T>& _ref;

public:
  T1Expr (T1<T>& a) : _ref(a) {}

  T& operator() (const size_t n) {
    return _ref(n);
  }

  T operator() (const size_t n) const {
    return _ref(n);
  }

  inline size_t dim1() const { return _ref.dim1(); }

  template<class B>
  inline const T1Expr<T1<T>, T, i>&
  operator=(const T1Expr<B, T, i>& b) {
    for (size_t k=0; k<_ref.dim1(); ++k)
      _ref(k) = b(k);

    return *this;
  }

  const T1Expr<T1<T>, T, i>&
  operator=(const T1Expr<T1<T>, T, i>& b) {
    for (size_t k=0; k<_ref.dim1(); ++k)
      _ref(k) = b(k);

    return *this;
  }
};
```

listato A.4: Specializzazione dell'espressione `T1Expr` nel caso in cui questa contenga un tensore di rango uno

Le tre classi introdotte necessitano di essere congiunte, per fare questo sovraccarichiamo l'operatore `T1::operator()(Index<i>& index)` imponendo che questo ritorni un'istanza di `T1Expr`

```
template<char i>
T1Expr<T1, T, i> operator()(Index<i>& index) {
  return T1Expr<T1, T, i>(*this);
}
```

cosí facendo riusciamo a creare espressioni a partire da un tensore

```
Index<'i'> i;
// un tensore di dimensione n...
T1<float> A (n);
// e la relativa espressione
A(i);
```

per poter effettuare operazioni tra espressioni dobbiamo creare delle specializzazioni parziali del template `T1Expr`, in paricolare siamo interessati al caso in cui l'espressione considerata sia un tensore (listato A.4).

Questa non è molto diversa dal template generico con alcune importanti differenze. Innanzitutto osserviamo che nella specializzazione non si copia l'oggetto contenuto, ma se conserva una referenza, poi viene concesso accesso bidirezionale agli elementi del tensore quando invece il modello generico permetteva solo di leggere l'elemento n-esimo dell'espressione, infatti in genere non ha senso allocare un elemento di un'espressione composta. Infine vengono definite due versioni dell'operatore = che permettono di assegnare l'espressione specializzata in funzione di una generica o di una specializzata.

In definitiva ora abbiamo a disposizione la sintassi che permette di assegnare un tensore:

```
Index<'i'> i;
// una coppia di tensori
T1<float> A (n);
T1<float> B (n);
...
// copiamo in A i valori presenti in B
A(i) = B(i);
```

Come anticipato l'indice effettua un controllo sul tipo statico, questa proprietà è illustrata nel seguente frammento di codice

```
Index<'i'> i;
Index<'j'> j;
// una coppia di tensori
T1<float> A (n);
T1<float> B (n);
...
// espressione non valida: i!=j
A(i) = B(j);
```

provare a copiare in `A` i valori presenti in `B` restituisce un errore in quanto il compilatore non è in grado di trovare un adeguato operatore per effettuare l'assegnazione. Questa proprietà risulta molto utile soprattutto quando si considerino espressioni complicate in quanto restituisce informazioni sulla loro correttezza sintattica.

Operazioni binarie

In una implementazione classica, le operazioni binarie sono ottenute sovraccaricando gli operatori aritmetici $*$, $+$, $-$, ... Nel caso dei tensori la moltiplicazione è essenzialmente ambigua e la tratteremo nello specifico quando si introdurranno le contrazioni interne ed esterne.

Consideriamo quindi in dettaglio la somma di due tensori, si vuole fornire un'interfaccia che restituisca la somma di espressioni che si riducano a tensori dello stesso rango:

$$w_i = a_i + b_i + c_i + \cdots ,$$

da un punto di vista logico il problema si divide in due operazioni, da un parte dobbiamo poter distinguere quale operazione effettuare poi dobbiamo costruire l'espressione che ne descriva il contenuto algebrico.

Come già visto i costrutti condizionali sono da implementarsi attraverso la specializzazione di operatori, nel caso considerato si deve specificare come debba agire l'operatore + quando gli argomenti siamo espressioni che rappresentano quantità riducibili a tensori di rango uno:

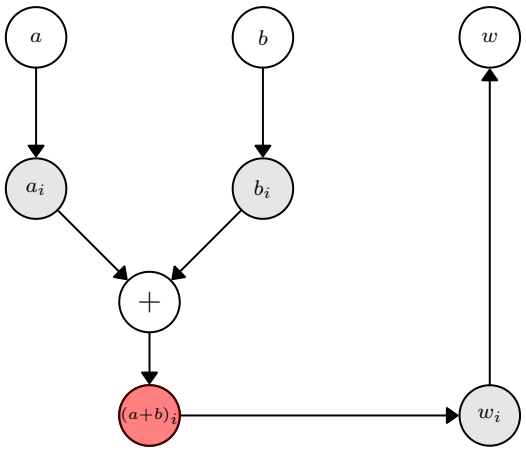


Figura A.1: implementazione dell'operazione $w = a + b$. A partire dagli oggetti a, b si costruiscono le espressioni a_i, b_i, attraverso la classe assistente viene istanziata l'espressione $(a + b)_i$ che può quindi essere uguagliata a w_i, quest'ultima espressione ha accesso agli elementi del tensore w producendo il risultato desiderato.

```
template<class A, class B, class T, char i>
class T1PlusT1 {
  const T1Expr<A, T, i>& _refA;
  const T1Expr<B, T, i>& _refB;

public:
  T operator() (const size_t n) const {
    return _refA(n) + _refB(n);
  }

  T1PlusT1 (const T1Expr<A, T, i>& a, const T1Expr<B, T, i>& b
      ) :
    _refA(a), _refB(b) {}

  size_t dim1() const { return _refA.dim1(); }
};
```

listato A.5: Classe assistente per l'espressione associata alla somma di due tensori di rango uno.

```
inline T1Expr<const T1PlusT1<A, B, T, i>, T,  i>
operator+ (const T1Expr<A, T, i>& a, const T1Expr<B, T, i>& b)
```

richiediamo che la somma in questo caso, restituisca la classe `T1Expr` istanziata per contenere il template `T1PlusT1` che andrà a descrivere operativamente l'operazione considerata. Osserviamo nuovamente che le due espressioni argomento dell'operatore devono essere parametrizzate dallo stesso indice, al solito si forza il compilatore ad effettuare un controllo statico sulla correttezza dell'espressione scritta.

La classe `T1PlusT1`, riprodotta nel listato A.5, rappresenta il risultato

```
template<class A, class B, class T, char i>
inline T1Expr<const T1PlusT1<A, B, T, i>, T,  i>
operator+ (const T1Expr<A, T, i>& a, const T1Expr<B, T, i>& b)
{
  typedef const T1PlusT1<A, B, T, i> Expr;

  return T1Expr<Expr, T, i>(Expr(a,b));
}
```

listato A.6: L'operatore semplicemente associa la classe assistente `T1PlusT1` alla relativa operazione.

dell'operazione considerata esposto mediante l'operatore `operator()` che infatti restituisce solo la somma degli elementi i-esimi dei tensori che contiene. Questi sono salvati in termini di referenze alle rispettive espressioni, la classe quindi non alloca risorse ed in effetti non effettua calcoli fino a quando non viene richiamata opportunamente.

Si è quindi implementata un'interfaccia che ricorda fedelmente la notazione di Einstein, comunemente utilizzata in fisica, per quanto riguarda la somma di tensori

```
Index<'i'> i;
T1<float> a (n), b (n), w(n);
...
w(i) = a(i) + b(i);
```

Il diagramma rappresentato in figura A.1 ne illustra il comportamento: il compilatore crea prima le istanze per le espressioni `a(i)`, `b(i)` e `w(i)`, cerca una specializzazione per l'operatore di somma binaria, se lo trova costruisce l'espressione ad esso associata

```
T1Expr<T1PlusT1<a,b,float,i>,float,i>
```

infine genera il codice che assegna all'espressione di sinistra quella di destra chiamando l'operatore `operator()`; solo a tempo di esecuzione vengono valutati ricorsivamente gli oggetti contenuti nelle espressioni, mentre le espressioni stesse sono risolte staticamente.

Contrazioni esterne ed interne

Implementare contrazioni tra indici ripetuti non presenta particolari difficoltà e procede essenzialmente come nel caso illustrato precedentemente, si vuole solo sottolineare il ruolo degli indici. Mentre precedentemente avevamo un solo indice, adesso stiamo considerando quantità del tipo

$$T_{ij}v_j, \quad w_i v_j, \quad t_{ik}z_{jk}, \quad \ldots$$

le relative espressioni sono selezionate, al solito, attraverso il meccanismo del controllo statico del tipo: affinché il compilatore possa elaborarle deve esistere una opportuna specializzazione dell'operatore di moltiplicazione per le contrazioni esterne e dell'operatore `operator()` per quelle interne.

Ad esempio il prodotto esterno di due tensori di rango un $a_i b_j$, costituisce un tensore di rango due, la specializzazione dell'operatore binario di moltiplicazione

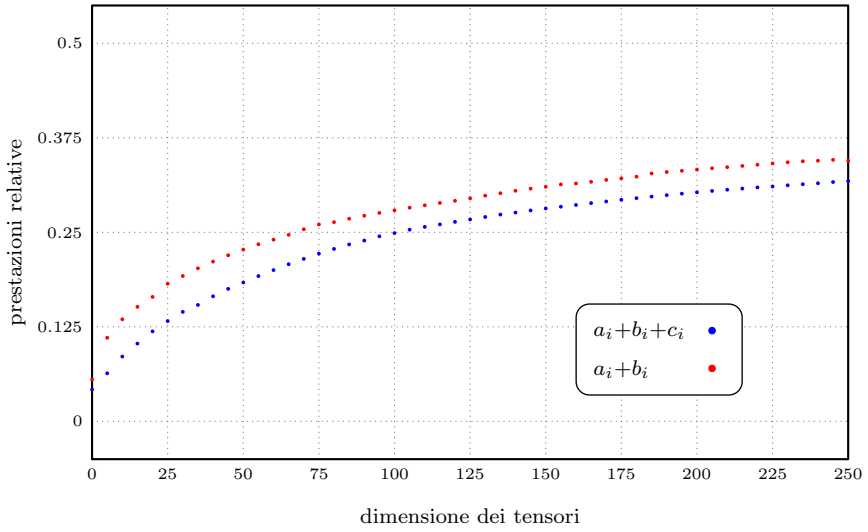


Figura A.2: analisi delle prestazioni nel caso della somma di tensori. Si confronta l'implementazione classica, ottenuta mediante il sovraccaricamento dell'operatore binario +, e il rispettivo template di espressioni. Nel grafico è riportato il rendimento relativo tra la seconda implementazione e la prima.

```
template<class A, class B, class T, char i, char j>
inline T2Expr<const T1TimesT1_T2<A, B, T, i, j>, T, i, j>
operator* (const T1Expr<A, T, i>& a, const T1Expr<B, T, j>& b)
{
    typedef const T1TimesT1_T2<A, B, T, i, j> Expr;

    return T2Expr<Expr, T, i, j>(Expr(a,b));
}
```

permette al compilatore di interpretare correttamente la sintassi

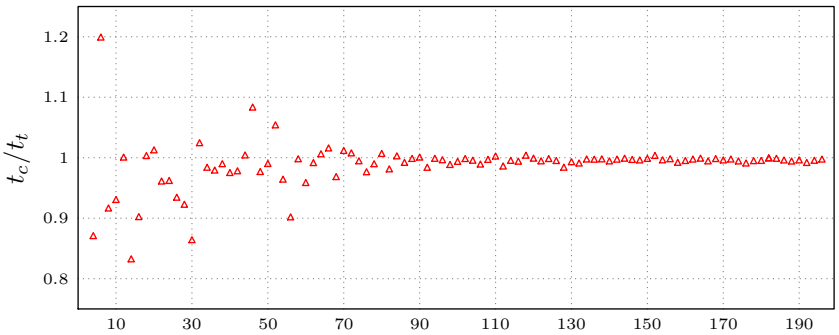
```
Index<'i'> i;
Index<'j'> j;
T1<float> a (n), b (n);
T2<float> T (n,n);
...
T(i,j) = a(i) * b(j);
```

e di produrre quindi il risultato voluto.

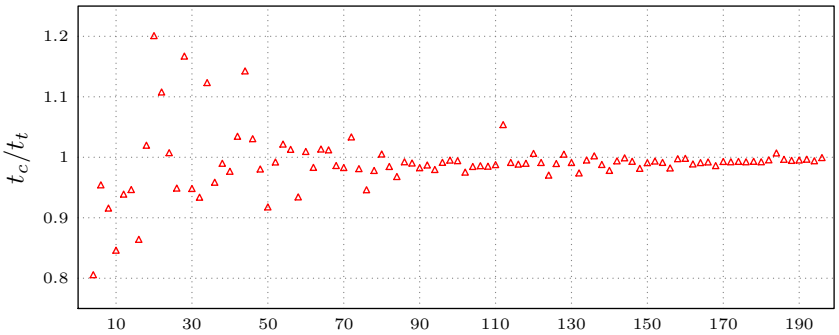
A.3 Un cenno alle prestazioni

Verifichiamo come l'implementazione offra, oltre all'evidente ergonomicità, prestazioni tali da renderla una alternativa efficace anche in contesti in cui le prestazioni rivestano un ruolo cruciale.

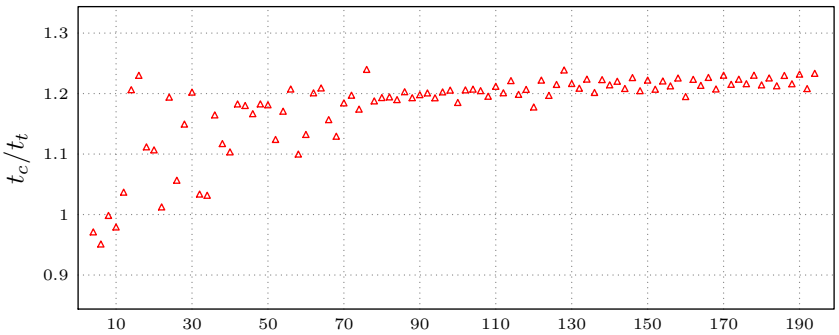
Confrontiamo inizialmente le espressioni template con i metodi tipici della programmazione orientata agli oggetti, consideriamo una classe che implementi la somma tra tensori sovraccaricando l'opportuno operatore binario in modo tale da avere un'interfaccia di facile utilizzo. Osserviamo che non è invece



(a) $w_{\mu\nu} = a_{\mu\eta} b_{\eta\nu}$



(b) $w_{\mu\nu} = a_{\mu\eta} b_{\eta\nu} + c_{\mu\nu}$



(c) $w_{\mu\nu} = a_{\mu\eta} b_{\eta\rho} c_{\rho\nu}$

Figura A.3: confronto tra le prestazioni nel caso in cui le operazioni siano calcolate con codice scritto a mano e con codice generato dai template di espressioni, nei grafici è riportato il rapporto tra il tempo t_c per il metodo classico e quello per i template di espressioni t_t al variare delle dimensioni dei tensori considerati.

possibile implementare contrazioni se non con metodologie lontani da questo spirito, ovvero definendo opportune funzioni o metodi.

Per come funziona il compilatore, e in certo senso il linguaggio, sovraccaricare gli operatori binari è una pratica poco efficiente. La semplice somma di due tensori

$$a_i = b_i + c_i,$$

illustra facilmente quanto detto,il compilatore infatti crea un'istanza temporanea della classe t_i, assegna a tale istanza il risultato della somma e quindi lo copia nella destinazione designata.

La scarsa efficienza è legata a due passaggi superflui, la creazione di un oggetto temporaneo e la necessità di scorrere due volte gli elementi del tensore. Osserviamo che mentre il primo è rilevante a piccole dimensioni, il secondo lo è per grandi dimensioni dei tensori. Come illustrato in precedenza, i template di espressioni sono stati studiati proprio per aggirare tali difficoltà come evidente dai risultati riportati in figura A.2.

Fornire prestazioni migliori rispetto ad un caso chiaramente inefficiente, non giustifica l'adozione dei template se non mostriamo che l'eventuale perdita di prestazioni rispetto al caso di codice ottimizzato manualmente sia ragionevole. Per loro natura i template sono valutati a tempo di compilazione, inoltre il profuso utilizzo dell'attributo `inline` per le espressioni, ci porta a credere che i risultati siano spesso equivalenti, i test riportati in figura A.3 concordano con queste considerazioni. Per espressioni semplici riusciamo a scrivere codice equivalente a quello generato dal compilatore come in A.3a e A.3b, quando le operazioni siano relativamente complesse è difficoltoso scrivere codice ottimizzato, i template di espressioni lasciano questo compito al compilatore ottenendo quindi prestazioni migliori come evidenziato dal caso (c).

Bibliografia

[1] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70(13):1895–1899, Mar 1993. v

[2] Lukasz Cincio, Jacek Dziarmaga, and Marek M. Rams. Multi-scale entanglement renormalization ansatz in two dimensions: Quantum ising model. *Physical Review Letters*, 100:240603, 2008. vi

[3] G. Evenbly and G. Vidal. Algorithms for entanglement renormalization. *Physical Review B*, 79:144108, 2009. v, 16, 17

[4] G. Evenbly and G. Vidal. Frustrated antiferromagnets with entanglement renormalization: ground state of the spin-1/2 heisenberg model on a kagome lattice, 2009. vi

[5] Glen Evenbly and Guifre Vidal. Entanglement renormalization in two spatial dimensions. *Physical Review Letters*, 102:180406, 2009. vi

[6] Yoshihiko Futamura. Partial evaluation of computation process, revisited, 1999. 38

[7] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965. 18

[8] Robert Koenig, Ben W. Reichardt, and Guifre Vidal. Exact entanglement renormalization for string-net models. *Physical Review B*, 79:195123, 2009. 12

[9] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, October 2000. v, 7

[10] Tobias J. Osborne and Michael A. Nielsen. Entanglement in a simple quantum phase transition. *Physical Review A*, 66(3):032110+, 2002. v, 30

[11] Robert N. C. Pfeifer, Glen Evenbly, and Guifre Vidal. Entanglement renormalization, scale invariance, and quantum criticality, 2008. 22

[12] D. Porras, F. Verstraete, and J. I. Cirac. Renormalization algorithm for the calculation of spectra of interacting quantum systems. *Physical Review B*, 73:014410, 2006. 12

[13] Matteo Rizzi, Simone Montangero, and Guifre' Vidal. Simulation of time evolution with the mera. *Physical Review A*, 77:052328, 2008. 17

[14] Subir Sachdev. *Quantum Phase Transitions*. Cambridge University Press, new edition edition, April 2001. 25

[15] Ulrich Schollwoeck. The density-matrix renormalization group. *Reviews of Modern Physics*, 77:259, 2005. v

[16] Erwin Unruh. Prime number computation, 1994. ANSI X3J16-94-0075/ISO WG21-462. 37

[17] T. L. Veldhuizen. C++ templates as partial evaluation, 1999. 38

[18] Todd L. Veldhuizen. Expression templates. *C++ Report*, 7(5):26–31, June 1995. Reprinted in C++ Gems, ed. Stanley Lippman. 39

[19] G. Vidal. Entanglement renormalization. *Physical Review Letters*, 99(22), 2007. v

[20] Steven R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69(19):2863–2866, Nov 1992. v, 2

[21] Kenneth G. Wilson. The Renormalization Group: Critical Phenomena and the Kondo Problem. *Rev. Mod. Phys.*, 47:773, 1975. 1